Desarrollo de una aplicación de aprendizaje adaptativo

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
Juan Luis López Javaloyes

Tutores:
Miguel Ángel Lozano Ortega
Rafael Molina Carmona

Septiembre 2021

AGRADECIMIENTOS

A mis profesores,

por haber sido capaces de transmitirme su pasión

A mis padres,

por no poner límites a mi búsqueda de la felicidad

A mis hermanos,

por descubrirme cada día cosas increíbles

A mi sobrina,

por demostrarme que la belleza está en las pequeñas cosas

A mis amigos,

por siempre fomentar mi demencia

Y a mi pareja,

por acompañarme en este viaje supersónico

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

El campo del aprendizaje online se encuentra en la actualidad en uno de sus mejores momentos. En oposición a la enseñanza tradicional en la que es el estudiante quien debe hacer un ejercicio activo de aproximación a las fuentes del conocimiento y de adaptación a su entorno, el desarrollo de las TIC y de las herramientas de enseñanza ubicua ha permitido acercar el proceso de aprendizaje al propio aprendiz, introduciéndose en sus rutinas y acomodándose a sus preferencias.

Desde la aparición en 2020 de la COVID-19, el aprendizaje online ha pasado de ser un medio complementario a ser una alternativa obligatoria a la enseñanza presencial. Los centros de formación se encontraron con la necesidad de continuar con el proceso de enseñanza a la vez que trataban de proteger la salud de todos aquellos involucrados en él (Daniel Hermawan, 2021), por lo que el foco de atención pasó rápidamente a otros medios en los que la interacción cara a cara se reducía al mínimo. Por suerte, la enseñanza virtual había evolucionado lo suficiente como para suplir esa necesidad con considerable éxito.

No obstante, parece evidente que la transformación tecnológica que conlleva el paso de la enseñanza presencial a la online no puede quedar únicamente en una modernización de las herramientas. Debe también suponer una revisión y actualización de los modelos y procesos de aprendizaje de manera que contribuyan potencialmente a la mejora del aprendizaje (Real-Fernández et al., 2017).

Con esta intención surge el modelo propuesto por Alberto Real, Rafael Molina y Faraón Llorens que, frente al aprendizaje tradicional de talla única, ofrece un aprendizaje adaptativo, colaborativo, flexible y escalable. Los objetivos de dicho modelo son, entre otros, permitir la personalización del contenido mediante la oferta de un itinerario de aprendizaje diferente para cada usuario, adaptar el aprendizaje al progreso de forma que el ritmo dependa de la cadencia en la que el usuario va superando competencias y ofrecer gran variedad de actividades, tanto en la forma en la que se realizan como en el tipo de conocimiento (Real-Fernández et al., 2017).

El proyecto *Adaptive Learning*, de la Unidad Científica de Innovación *Ars Innovatio*, se basa en este modelo para desarrollar una plataforma homónima en la que se ofrecen cursos que se caracterizan por ser personalizados, adaptativos y basados en competencias y actividades de aprendizaje. En ella se pueden encontrar diferentes

espacios de trabajo en función del rol de usuario: un espacio de creación de contenidos, un espacio para la docencia y un espacio para el aprendizaje.

A pesar de la potente herramienta educativa que supone esta plataforma, parece necesario dotar al sistema de una flexibilidad todavía mayor. Por esta razón en el proyecto actual se propone el desarrollo de una aplicación móvil que adapte la versión web de la plataforma *Adaptive Learning* a estos dispositivos y que potencie sus funcionalidades, permitiendo que el usuario final pueda acceder a cualquier contenido en cualquier instante y desde cualquier lugar que él desee.

El documento actual recoge el proceso de creación de esta aplicación, siguiendo una estructura que pretende facilitar la comprensión de todo el trabajo realizado.

En primer lugar, se encuentra el apartado “Marco teórico”. En él se revisa brevemente el estado actual del campo del aprendizaje online, se establece el punto desde el que parte el desarrollo del proyecto y se hace un breve estudio de las principales aplicaciones disponibles con características similares a la que se va a desarrollar.

El siguiente apartado, “Objetivos”, recoge, como su nombre indica, los objetivos perseguidos en el desarrollo de este proyecto. Se describe primero la intención principal y después se listan los diferentes aspectos relativos al producto que se desean alcanzar.

El apartado que hay a continuación es “Metodología”. En él se describe la metodología de desarrollo empleada a lo largo del proyecto, mostrando las funcionalidades aportadas en cada iteración y su coste en tiempo.

Seguidamente se encuentra el “Cuerpo del trabajo”. Se trata del apartado que contiene el grueso de la memoria. En él se detalla cada uno de los pasos del ciclo de vida de nuestro proyecto: el análisis, el diseño y la implementación.

El siguiente apartado de este documento es el de las “Conclusiones”. En él se revisan los objetivos, comparando las intenciones al inicio del proyecto con el trabajo que finalmente se ha realizado, y se describen las posibles mejoras y nuevos desarrollos que podrían implementarse en un futuro.

Por último, encontramos el apartado de la “Bibliografía”. En él se citan todas aquellas fuentes de las que se ha extraído información para el desarrollo del producto y para la elaboración de este documento.

2. MARCO TEÓRICO

En este apartado se realiza en primer lugar un estudio del estado actual de la disciplina del aprendizaje virtual, haciendo un breve repaso de las metodologías que son tendencia en los últimos años. Seguidamente se presenta el estado actual del proyecto *Adaptive Learning* para a continuación describir las mejoras que se desean aportar con la realización de la aplicación móvil. Por último, se revisan las principales aplicaciones con funcionalidades similares que actualmente están en el mercado.

2.1. Tendencias *e-Learning*

El *e-Learning* o aprendizaje virtual es un sistema de enseñanza y aprendizaje que se basa en el uso de Internet, así como de dispositivos y herramientas que se puedan conectar a esta red (Cross, 2004). Esta modalidad formativa supone una evolución respecto a la formación a distancia debido a que permite el aprendizaje desde cualquier lugar con el único requerimiento de disponer de un dispositivo con conexión a internet.

Las principales ventajas de esta modalidad de aprendizaje se pueden resumir en los siguientes puntos (Rodenes Adam et al., 2013):

- Centrada en el alumno y a su propio ritmo, personalizando el aprendizaje en función de las preferencias de cada uno
- Facilita el acceso al aprendizaje, ofreciendo una solución a las restricciones de tiempo y lugar
- Eficaz en coste para el alumno
- Potencialmente disponible para una audiencia global
- Acceso ilimitado al conocimiento
- Capacidad de archivo para reutilizar y compartir el conocimiento
- Permite una supervisión individual más precisa y continuada
- Crea comunidad social y cooperación
- Mejora la comprensión y la retención por el uso del multimedia, reduciendo así los tiempos de aprendizaje
- Facilita la actualización de la información y de los contenidos

El aprendizaje virtual se está consolidando como uno de los principales métodos para estudiar cualquier tipo de formación, principalmente por su amplia y variada oferta formativa y su flexibilidad horaria.

Se pueden encontrar diferentes metodologías que hacen uso o se benefician del uso del *e-Learning*. A continuación, se enumeran algunas de aquellas que están más en tendencia los últimos años (Davis, 2020) (EUCIM, 2021):

- ***Flipped Classroom***: es un tipo de aprendizaje mixto en el que se propone que los alumnos estudien y preparen las lecciones fuera de clase, accediendo en casa a los contenidos de las asignaturas para que, posteriormente, sea en el aula donde hagan los deberes, interactúen y realicen actividades más participativas.

- ***Microlearning***: metodología que surge de la adaptación de la formación tradicional a la sociedad del consumo rápido del "aquí y ahora". Esta modalidad se organiza en torno a "píldoras de aprendizaje", es decir, pequeños módulos formativos con una duración que no debe superar los 30 minutos.

- **Gamificación**: metodología que hace referencia al uso de estrategias propias de los juegos en entornos no lúdicos, lo que ayuda a enseñar habilidades colaborativas, generar competitividad y permitir la experimentación. Puede proveer de una inmediata aplicación de los materiales e interacción con los mismos. Conforme aumenta la comprensión también lo hace el compromiso, la retención y las notas del estudiante.

- **Aprendizaje Condicional o Adaptativo**: metodología en la que el creador de los materiales formativos define "a priori" diferentes itinerarios a través de los contenidos. La selección del itinerario para cada estudiante depende de sus resultados en diferentes pruebas de nivel intermedias que se le proponen a lo largo del curso. Utiliza un conjunto de técnicas encaminadas a ofrecer al estudiante online una experiencia personal y diferenciada, partiendo de la base de que cada alumno es diferente y único, ya que posee una experiencia, unas necesidades formativas o un estilo de aprendizaje diferentes.

- ***m-Learning***: El *m-Learning*, o *Mobile Learning*, es una extensión del *e-Learning* que ofrece una mayor flexibilidad, ayudando a que el alumno cuente con oportunidades para aprender en cualquier sitio y en el momento deseado y con una oferta de contenidos más ligeros y de sesiones diseñadas para durar pocos minutos.

2.2. Estado actual del proyecto *Adaptive Learning*

Adaptive Learning es un proyecto desarrollado por el Laboratorio de Innovación en Educación de la Unidad Científica de Innovación Empresarial Ars Innovatio, que propone el diseño y desarrollo de una **plataforma inteligente de aprendizaje adaptativo**. Los cursos, diseñados en dicha plataforma por docentes con los materiales formativos adecuados, disponen idealmente de las siguientes características:

- **Personalizados**: se adaptan a las características e intereses de cada aprendiz.
- **Adaptativos**: se adaptan de forma dinámica a la situación del aprendiz en cada momento.
- **Basados en competencias** y **actividades**: desarrollan las competencias, destrezas y habilidades que propone el curso a través de actividades de aprendizaje.

La metodología de esta plataforma se basa en el modelo de aprendizaje propuesto por Alberto Real, Rafael Molina y Faraón Llorens (Real-Fernández et al., 2017) que, frente al aprendizaje tradicional de talla única, ofrece un aprendizaje adaptativo, colaborativo, flexible y escalable. Los elementos centrales de este modelo son:

- **Competencias**: combinación de conocimientos, destrezas y actitudes necesarias para una acción eficaz ante un determinado problema.
- **Actividades** de aprendizaje: una acción o tarea que lleva a los estudiantes que la realizan a desarrollar una o varias competencias y, por tanto, a aprender.

Además, se propone una estructura formada por tres elementos principales:

- El **cuadro de mando del docente**, para el diseño del curso en base a competencias y actividades.
- El **espacio de trabajo del estudiante**, en el que se realizan las actividades formativas y se mantiene el estado de competencias y actividades.
- El **motor de selección**, encargado de seleccionar las actividades en función del progreso del estudiante en su aprendizaje.

De esta manera se permite la personalización del contenido, adaptado al nivel de conocimientos de cada usuario y a su progreso, y a través de itinerarios de aprendizaje diferentes elegidos por el propio usuario, incorporando conceptos de refresco y refuerzo y la posibilidad de elegir para dotar a los estudiantes de autonomía.

Actualmente la plataforma *Adaptive Learning* se encuentra activa y en funcionamiento. Se puede acceder a ella con diferentes **roles** para desempeñar diferentes tareas:

- **Diseñador**:
    - Crear actividades de diferentes tipos, indicando las soluciones para que la plataforma evalúe automáticamente al aprendiz cuando éste las realice.
    - Crear competencias, que servirán posteriormente para confeccionar los cursos.
    - Relacionar actividades y competencias para indicar qué competencias desarrolla cada actividad.
- **Docente**:
    - Crear cursos mediante la confección de mapas de competencias, pudiendo formar diferentes itinerarios de aprendizaje que el aprendiz podrá seguir una vez comience el curso.
    - Añadir docentes y aprendices.
    - Invitar a docentes y aprendices mediante el envío de un correo de invitación.
    - Supervisar el progreso de los aprendices, pudiendo visualizar la puntuación obtenida en cada competencia y actividad y pudiendo revisar las respuestas que ha dado en las actividades realizadas por los mismos.
- **Aprendiz**:
    - Acceder a su progreso en el curso, visualizando en el mapa de competencias la puntuación que ha obtenido en cada una, cuáles están disponibles y cuáles bloqueadas y la puntuación que ha de obtener en una competencia para desbloquear otra de la que depende.
    - Realizar actividades de aquellas competencias que tenga desbloqueadas y obtener una puntuación inmediata en cuanto envía su respuesta.

Adaptive Learning trabaja en estos momentos haciendo uso de un motor de selección básico por el cual se le ofrece al estudiante una actividad aleatoria de todas las disponibles para él en una determinada competencia, pero en el futuro se desarrollará el algoritmo para hacer una selección adaptada al uso y al perfil del estudiante.

La plataforma dispone en la actualidad de al menos 4 cursos reales con 101 aprendices, 47 competencias y 694 actividades de aprendizaje diferentes.

2.3. Propuesta de mejora: *m-Learning*

Aunque la plataforma *Adaptive Learning* ofrece en la actualidad una metodología de aprendizaje que mejora notablemente los modelos educativos tradicionales, tal y como se ha expuesto anteriormente, parece interesante dotar al sistema de una mayor flexibilidad para que el usuario final pueda acceder a cualquier contenido en cualquier instante y desde cualquier lugar. El interés del uso del móvil como una herramienta de aprendizaje queda demostrado por el amplio conocimiento y aceptación de esta tecnología por parte de los estudiantes (Sarrab et al., 2015).

Para ello se propone crear una aplicación móvil, extensión de la plataforma web, que añada **beneficios** propios del ***m-Learning***, como son:

- **Sin barreras físicas o temporales**: la posibilidad de utilizar dispositivos portátiles, como móviles o tablets, ampliamente utilizados en la actualidad, permite a los estudiantes disponer de los contenidos siempre que quieran y desde donde ellos quieran. De esta manera el aprendizaje se consigue ajustar a las necesidades del alumno en cada momento (nubemia, 2015).
- **Variedad de recursos didácticos**: los formatos en los que los estudiantes pueden acceder a la información son muy diversos, y la interactividad que ofrece el uso de los sensores y de las funcionalidades de estos dispositivos mejoran considerablemente la experiencia de los usuarios (nubemia, 2015).
- **Aprendizaje contextualizado e informal**: la posibilidad de integrar el aprendizaje en la rutina diaria del alumno posibilita que lo contextualice de manera natural y adquiera conocimientos y habilidades dentro de su entorno más inmediato. El estudiante ya no necesita de un horario estructurado y formal, sino que el aprendizaje forma parte de su vida cotidiana (tekman education, 2021).
- **Interacción inmediata**: gracias a la posibilidad de automatizar ciertos procesos y al uso de los sistemas de mensajería instantánea y notificaciones, el *feedback* que se produce es mucho más veloz (tekman education, 2021).

Además, como aplicación móvil de aprendizaje, se han detectado una serie de **características** de las que la app debería disponer para ofrecer la mejor experiencia posible:

- **Interfaz de usuario intuitiva** que permita al usuario acceder a antiguos y nuevos cursos con unos pocos clics y que disponga de una navegación con únicamente

botones esenciales de manera que no parezca abarrotado y distraiga a los aprendices.

- **Notificaciones *Push*** que faciliten la apertura de un canal de comunicación entre los aprendices y los docentes y que supongan una forma de mantener la atención de los primeros.
- **Gamificación**: los elementos de gamificación añaden sabor y diversión a la experiencia de aprendizaje, animando a los aprendices a interactuar más con el entrenamiento y aumentando su motivación.
- **Soporte de contenidos multimedia**: la app debería soportar diversos tipos de contenidos para conseguir una atracción mayor, así como un aprendizaje mejorado.
- **Sincronización del progreso**: los aprendices deberían poder utilizar tanto la plataforma web como la app de forma intercambiable sin perder el progreso que hayan obtenido en alguna de ellas (eLearning Industry, 2020).

2.4. Estudio de la competencia

A continuación, se procede a hacer un repaso de las principales aplicaciones de aprendizaje disponibles que ofrecen un servicio semejante al que se desea proporcionar con este proyecto. Para concluir el apartado, se realiza una breve comparación de éstas con la app de Adaptive Learning.

Udemy

Ilustración 1: Logo de Udemy.

Udemy dispone de una app de aprendizaje que cuenta con más de 5500 cursos con la mejor calificación que ofrece al usuario la posibilidad de adquirir nuevas habilidades en cualquier momento y lugar. Permite descargar los cursos para continuar con el aprendizaje incluso sin conexión a internet o con conexión inestable, y proporciona la posibilidad de conectarse a Chromecast para ver los cursos en televisión. Dispone de notificaciones *PUSH* para personalizar recordatorios de aprendizaje que se adapten al

horario del usuario y permite tomar notas y añadir marcadores que ayudan a recordar lo aprendido. En cuanto a contenido, ofrece clases en vídeo, audio y de texto, y para reforzar el aprendizaje presenta cuestionarios que el usuario debe completar. Además, pone al alcance de los alumnos la posibilidad de enviar preguntas a los instructores para ampliar conocimientos u obtener ayuda (Udemy, 2021).

LinkedIn Learning

Ilustración 2: Logo de LinkedIn Learnin.

LinkedIn Learning es una app de aprendizaje que ayuda a las organizaciones a aumentar el compromiso del alumno y a desarrollar las competencias que estos necesitan. Incluye cursos sobre negocios, tecnología y aspectos recreativos, ofreciendo recomendaciones personalizadas de cursos según su perfil de la plataforma. También permite la descarga de los cursos para acceder a ellos sin conexión, así como el guardado o marcado de cursos para verlos más tarde. Una vez que se completa un curso este aparece en el perfil de LinkedIn del usuario que lo realiza (LinkedIn Learning, 2021).

Khan Academy

Ilustración 3: Logo de Khan Academy.

Khan Academy dispone de una app para aprender a través de materiales multimedia materias como cálculo, álgebra, química, biología, astronomía, finanzas, etc. y permite que cualquier persona de cualquier país tenga acceso a la educación. Los cursos de Khan Academy pueden estar formados por ejercicios interactivos, vídeos, artículos, etc. Las habilidades se mejoran realizando ejercicios, pruebas y exámenes con retroalimentación inmediata y pistas paso a paso. Además, el contenido se puede marcar y descargar para acceder a él sin conexión a internet. En todo momento el progreso se

sincroniza con la web khanacademy.org para no perder el avance (Khan Academy, 2021).

Duolingo

Ilustración 4: Logo de Duolingo.

Duolingo es una app de aprendizaje de idiomas con una alta gamificación: completar unidades permite avanzar en el curso, equivocarse en las respuestas de los ejercicios disminuye el número de vidas del usuario y al realizar actividades se ganan puntos e incluso se sube de nivel (Duolingo, 2021).

EdX

Ilustración 5: Logo de EdX.

La app de aprendizaje de EdX se basa en la plataforma del mismo nombre fundada por profesores de Harvard y MIT, y cuenta con más de 34 millones de aprendices. Ofrece clases online en directo sobre ciencia de datos, *python*, etc. que permiten obtener certificados o créditos universitarios. Los cursos están disponibles para descargar y poder acceder a ellos en cualquier momento sin necesidad de acudir a un aula. La evaluación de los conocimientos se lleva a cabo mediante la realización de cuestionarios y exámenes a medida que se avanza en cada curso. Además, ofrece la posibilidad de visualizar anuncios y hojas informativas de los cursos (edX, 2021).

Coursera

Ilustración 6: Logo de Coursera.

Coursera ofrece más de 4000 cursos online asociados con universidades de prestigio y organizaciones. Los vídeos de clases en línea de los que dispone son descargables, lo que permite verlos en cualquier momento. Además habilita la opción de recibir notificaciones de próximos cursos. Como en otras aplicaciones, el progreso se sincroniza con el sitio web, en este caso Coursera.org. Los cursos están disponibles en al menos 12 idiomas y ofrecen subtítulos en el idioma nativo y en inglés (Coursera, 2021).

Codecademy

Ilustración 7: Logo de Codecademy Go.

Codecademy ofrece una app de aprendizaje de programación que facilita la captación de los conceptos principales de la escritura de código. Ofrece tarjetas didácticas diariamente que se pueden ojear de forma rápida. Además, cada lección tiene un ejercicio o práctica al final para familiarizarse mejor con lo que se ha aprendido en la lección. Dispone también de una herramienta para tomarse descansos, un temporizador personalizable que te recuerda cuándo iniciar un descanso y cuándo retomar la práctica (Codecademy, 2021).

Nuestro proyecto (Adaptive Learning)

Para concluir el estudio de la competencia, cabe mencionar que la aplicación *Adaptive Learning* comparte muchas características con las principales aplicaciones de aprendizaje que existen en el mercado actualmente, pero también dispone de ciertos elementos diferenciadores que la hacen destacar sobre ellas.

Al igual que muchas de las otras propuestas, *Adaptive Learning* sincroniza el progreso con el sitio web para que el usuario no pierda su avance. También hace uso de notificaciones para captar y mantener la atención de este y dispone de una interfaz agradable que pretende mejorar su experiencia.

El principal elemento que la diferencia de las demás es que permite la creación de cursos de cualquier tipo de temática o campo sin ningún tipo de limitación, pues los contenidos de un curso dependen totalmente de los propios usuarios que lo han diseñado. Además, el aprendizaje se realiza mediante píldoras de aprendizaje en la propia actividad, y aporta características de gamificación al progreso del usuario sin tener que limitarse al campo de los idiomas.

En el futuro, se diferenciará todavía más de la competencia por la utilización de una IA en el motor de selección de actividades, que conseguirá determinar cuál es la actividad más adecuada para un determinado aprendiz en un instante concreto en función del uso que haya hecho anteriormente de la plataforma, de sus hábitos de estudio y sus preferencias.

Por lo tanto, la aplicación *Adaptive Learning* pretende ser una combinación perfecta de las características más importantes de las principales tendencias en aprendizaje online que existen actualmente en el mercado, aportando aspectos de *microlearning*, gamificación, *m-learning* y, por supuesto, aprendizaje adaptativo.

3. OBJETIVOS

En este apartado se detalla el propósito perseguido con la realización del proyecto y se concretan los objetivos específicos que se desean conseguir.

3.1. Objetivo general

El objetivo de este proyecto es la **creación de una aplicación móvil de aprendizaje** que sirva como **extensión de la plataforma web** creada en el proyecto ***Adaptive Learning***. Se adaptará para ello la metodología al nuevo formato móvil y se aportará valor añadido mediante el aprovechamiento de las funcionalidades que ofrece este tipo de dispositivos.

Se aplicarán por lo tanto los conocimientos extraídos del estudio realizado anteriormente sobre aplicaciones móviles de aprendizaje, así como los diversos conocimientos sobre programación móvil adquiridos en el Máster, con la intención de obtener un producto usable, atractivo y útil para el usuario.

3.2. Objetivos específicos

Con el desarrollo de la aplicación *Adaptive Learning* se desea alcanzar los siguientes objetivos:

- **Implementar el acceso de aprendices a cursos** en los que puedan desarrollar competencias en un mapa no necesariamente lineal, permitiéndoles la elección de su propio recorrido de aprendizaje por él. **Facilitarles la realización de las actividades de aprendizaje** que desarrollan dichas competencias, ofreciéndoselas secuencialmente y de forma sencilla y atractiva.
- **Implementar el acceso de docentes** a los cursos que imparten para la **evaluación y revisión** de las aptitudes de los aprendices, permitiéndoles visualizar su progreso y el resultado de las actividades que han realizado.
- **Implementar el acceso de diseñadores** a la **visualización, creación, edición y borrado de los contenidos** que formarán los cursos: competencias, actividades, etc.
- **Implementar la comunicación entre la plataforma Android y un servicio web** mediante una interfaz tipo *REST*.
- **Integrar la aplicación con servicios de notificaciones** proporcionados por las plataformas móviles de Android.

- **Aprovechar el hardware del dispositivo para mejorar la usabilidad** de la aplicación respecto a la versión web.
- **Practicar la creación de aplicaciones en Android Studio** mediante la implementación de una app nativa completa.

4. METODOLOGÍA

En este apartado se hace una breve descripción de la metodología de desarrollo empleada durante el proyecto.

Dado que se trata de un proyecto individual, el desarrollo del mismo no tiene grandes requerimientos de planificación y control. La mayor preocupación y recurso a controlar es el tiempo, pues el trabajo tiene un alcance inicial bastante ambicioso que ha de poder cumplirse en un periodo determinado.

Se ha escogido por lo tanto una **metodología ágil de desarrollo incremental, *extreme programming***, según la cual las tareas a realizar se han agrupado en pequeños bloques temporales (iteraciones) que conducen a un determinado prototipo. La aplicación evoluciona a partir de los resultados de las iteraciones anteriores, añadiendo nuevos objetivos y requisitos a la vez que se mejoran los que ya fueron completados.

Cada iteración tiene una duración de dos semanas y al final de ese periodo se obtiene un prototipo mejorado que ha de poder funcionar correctamente para ser mostrado al cliente (en este caso, los tutores del proyecto).

El esfuerzo que requiere cada tarea a realizar se estima utilizando como medida el punto. Un punto equivale a una semana ideal de programación, o lo que es lo mismo, 25 horas aproximadas de trabajo (5 por día laboral). Por lo tanto, teniendo en cuenta que contamos con 4 meses para el desarrollo, se conoce que se pueden completar hasta 16 puntos. En consecuencia, tras calcular los esfuerzos para cada tarea, se han planificado 7 iteraciones de aproximadamente 2 puntos cada una, a lo largo de los 4 meses.

A continuación se especifican estas iteraciones, indicando el prototipo que se propone obtener al final de cada una y las tareas que la componen con sus esfuerzos.

Iteración 1: Documento de análisis y diseño y app con autenticación de usuarios

- Estudio del marco teórico *(0.32 puntos – 8 h)*
- Definición de objetivos *(0.16 puntos – 4 h)*
- Especificación de requisitos *(0.24 puntos – 5 h)*
- Especificación del diseño *(0.24 puntos – 6 h)*
- Creación del proyecto en Android Studio *(0.04 puntos – 1 h)*
- Implementación de la *splash screen (0.12 puntos – 3 h)*

- Creación del modelo de Usuario *(0.04 puntos – 1 h)*
- Creación de la clase ayuda para hacer peticiones HTTP *(0.24 puntos – 6 h)*
- Implementación de la vista y la lógica de inicio de sesión *(0.56 puntos – 14 h)*

Puntos: 1.96 puntos. Horas empleadas: **49 h**.

Reunión para validación del prototipo con los tutores: 16 de abril de 2020

Iteración 2: App con registro de usuarios y configuración de perfil y preferencias

- Implementación de la vista y la lógica de registro *(0.64 puntos – 16 h)*
- Creación de actividad base con *toolbar* de sesión iniciada *(0.24 puntos – 6 h)*
- Implementación de la vista y la lógica para actualizar datos de perfil *(0.4 puntos – 10 h)*
- Implementación de la creación de QR de usuario *(0.12 puntos – 3 h)*
- Implementación de la vista y la lógica para actualizar contraseña *(0.28 puntos – 7 h)*
- Implementación de la vista y la lógica para cambiar idioma *(0.36 puntos – 9 h)*

Puntos: 2.04 puntos. Horas empleadas: **51 h**.

Reunión para validación del prototipo con los tutores: 30 de abril de 2020

Iteración 3: App con diseñador básico

- Creación de vista paginada *(0.36 puntos – 9 h)*
- Creación de modelos de Cursos, Competencias y Actividades *(0.16 puntos – 4 h)*
- Implementación del listado de cursos diseñados *(0.32 puntos – 8 h)*
- Implementación del listado de competencias diseñadas *(0.16 puntos – 4 h)*
- Implementación del listado de actividades diseñadas *(0.16 puntos – 4 h)*
- Implementación del borrado de cursos, competencias y actividades diseñadas *(0.28 puntos – 7 h)*
- Implementación de vista y lógica para la creación/edición de competencias *(0.52 puntos – 13 h)*

Puntos: 1.98 puntos. Horas empleadas: **49 h**.

Reunión para validación del prototipo con los tutores: 14 de mayo de 2020

Iteración 4: App con diseñador avanzado

- Implementación de vista y lógica para la creación/edición de actividades *(1 puntos – 25 h)*
- Implementación de vista y lógica para la creación/edición de cursos *(1.04 puntos – 28 h)*

Puntos: 2.12 puntos. Horas empleadas: **53 h**.

Reunión para validación del prototipo con los tutores: 28 de mayo de 2020

Iteración 5: App con visualización básica de cursos para aprendices

- Implementación de vista y lógica para representar una competencia de curso *(0.56 puntos – 14 h)*
- Implementación de la vista y lógica base del curso de un aprendiz *(0.44 puntos – 11 h)*
- Implementación de vista y lógica del listado de competencias del aprendiz en un curso *(0.52 puntos – 13 h)*
- Implementación de vista y lógica del mapa de competencias del aprendiz en un curso *(0.6 puntos – 15 h)*

Puntos: 2.12 puntos. Horas empleadas: **53 h**.

Reunión para validación del prototipo con los tutores: 11 de junio de 2020

Iteración 6: App con visualización completa de cursos y realización de actividades para aprendices

- Implementación de vista y lógica de la información de una competencia de curso *(0.52 puntos – 13 h)*
- Implementación de vista y lógica de la realización de una actividad *(0.64 puntos – 16 h)*
- Implementación del *feedback* que recibe el aprendiz al contestar la actividad *(0.32 puntos – 8 h)*
- Implementación de vista y lógica de la transición entre dos actividades *(0.56 puntos – 14 h)*

Puntos: 2.04 puntos. Horas empleadas: **51 h**.

Reunión para validación del prototipo con los tutores: 25 de junio de 2020

Iteración 6: App con gestión docente de un curso

- Implementación de vista y lógica de listado de aprendices y su progreso general *(0.4 puntos – 10 h)*
- Implementación de vista y lógica del progreso completo de un aprendiz en el curso *(0.56 puntos – 14 h)*
- Implementación de vista y lógica de una actividad realizada por el aprendiz *(0.24 puntos – 5 h)*
- Implementación de vista y lógica de gestión de aprendices de un curso *(0.28 puntos – 7 h)*
- Implementación de vista y lógica de gestión de docentes de un curso *(0.24 puntos – 6 h)*
- Implementación de vista y lógica de gestión de invitaciones de un curso *(0.24 puntos – 6 h)*

Puntos: 1.92 puntos. Horas empleadas: **48 h**.

Reunión para validación del prototipo con los tutores: 9 de julio de 2021

Iteración 7: App completa

- Detección de errores *(0.6 puntos – 15 h)*
- Corrección de errores *(1.24 punto – 31 h)*

Puntos: 1.84 puntos. Horas empleadas: **46 h**.

Reunión para validación del prototipo con los tutores: 23 de julio de 2021

5. CUERPO DEL TRABAJO

En este apartado se concentra el grueso del trabajo, realizando un recorrido por los principales pasos del ciclo de vida de un proyecto: el análisis, el diseño y la implementación.

5.1. Análisis y especificación

A continuación, se procede a hacer una recopilación y especificación de las características funcionales y no funcionales que deberá cumplir la aplicación a desarrollar.

5.1.1. Actores del sistema

Para comenzar el análisis es necesario especificar qué usuarios interactuarán con la aplicación y cuáles serán las principales funcionalidades a las que podrán acceder. Los actores de nuestro sistema son:

- **Usuario no autenticado**: actor con capacidades muy limitadas. Únicamente podrá acceder al inicio de sesión y al registro de nuevo usuario, ya que la aplicación contará con un contenido público muy limitado.
- **Usuario autenticado:** actor que puede acceder a la plataforma mediante el inicio de sesión. Las capacidades de las que dispone dependen del rol que tenga:
    - **Diseñador**: es el actor que generará el contenido. Podrá crear nuevas actividades, nuevas competencias y nuevos cursos, así como relacionar estos elementos entre sí con el objetivo de confeccionar unidades de aprendizaje de utilidad para los aprendices a los que se dirigirá cada curso.
    - **Docente**: se trata del actor que se encargará de vigilar el progreso de los aprendices en los cursos. Como docente de un curso, podrá añadir y eliminar usuarios, acceder a un resumen del estado del curso, visualizar el progreso de cada aprendiz, así como revisar sus actividades realizadas.
    - **Aprendiz**: actor que accederá al contenido para aprender. Una vez añadido a un curso, este actor podrá visualizar el mapa de competencias de este, acceder a la realización de actividades de aquellas competencias desbloqueadas para él, así como visualizar su progreso.

5.1.2. Casos de uso

Los casos de uso son diagramas que describen las actividades que realiza alguien o algo en un sistema para llevar a cabo algún proceso o acción que da lugar a un resultado de valor observable. A continuación, se procede a definir los casos de uso de la aplicación a desarrollar en función del actor que interviene en cada proceso.

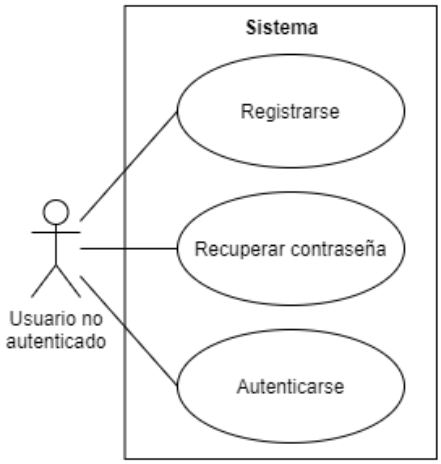

Ilustración 8: Usuario no autenticado - Casos de uso [Gráfico]

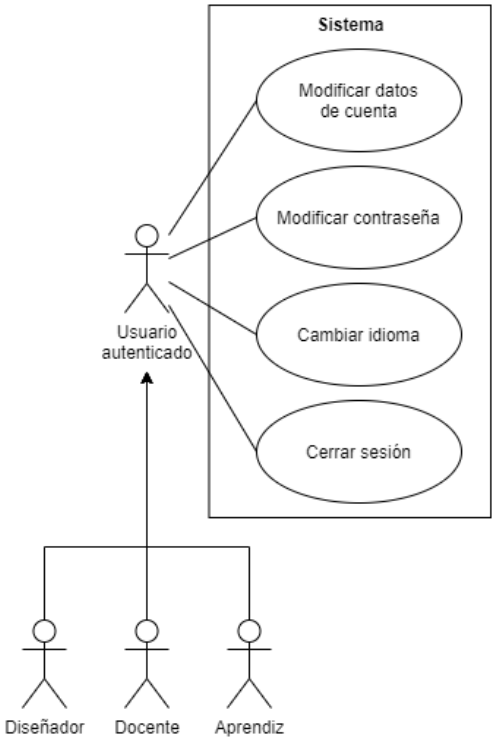

Ilustración 9: Usuario autenticado (genérico) - Casos de uso.

Ilustración 10: Usuario autenticado (diseñador) - Casos de uso.

Ilustración 11: Usuario autenticado (docente) - Casos de uso.

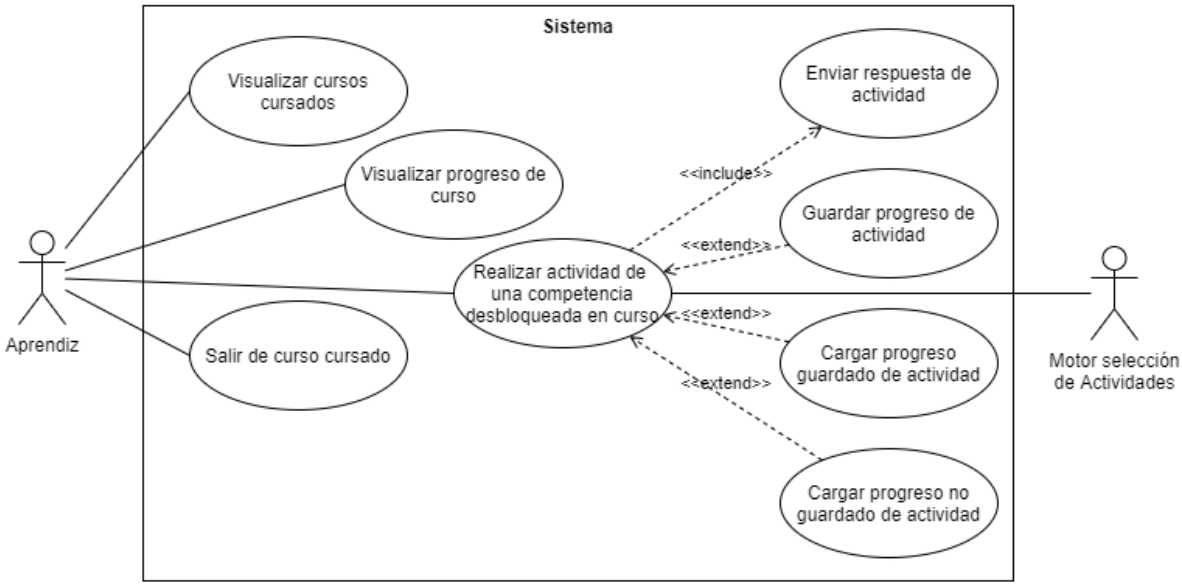

Ilustración 12: Usuario autenticado (aprendiz) - Casos de uso.

5.1.3. Análisis de requisitos

Un análisis de requisitos es el proceso de definición de las expectativas del usuario hacia un nuevo software a desarrollar, recopilando y definiendo las necesidades que se deben cumplir en dicho producto (Visual Paradigm, 2021). En este apartado se procede a concretar dichas definiciones.

5.1.3.1. Requisitos funcionales

Dentro del análisis de requisitos, los requisitos funcionales tienen el objetivo de expresar la naturaleza del funcionamiento del sistema mediante la declaración de los servicios que prestará.

Los requisitos funcionales de este proyecto se han divido según el actor principal involucrado: usuario no autenticado, usuario autenticado (genérico), usuario autenticado diseñador, usuario autenticado docente y usuario autenticado aprendiz. Todos estos requisitos se han detallado en el Anexo 1: Especificación de requisitos funcionales. De ellos, podríamos destacar como más importantes los siguientes:

- Registro (Usuario no autenticado)
- Inicio de sesión (Usuario no autenticado)
- Cierre de sesión (Usuario autenticado)
- Diseño de actividad (Usuario autenticado diseñador)
- Diseño de competencia (Usuario autenticado diseñador)
- Vinculación de una competencia con actividades (Usuario autenticado diseñador)
- Diseño de cursos (Usuario autenticado diseñador)
- Adición de competencias a un curso (Usuario autenticado diseñador)
- Adición de dependencias entre competencias de un curso (Usuario autenticado diseñador)
- Adición de usuarios como aprendices de un curso (Usuario autenticado docente)
- Visualización del progreso de un aprendiz en un curso (Usuario autenticado docente)
- Visualización de actividad realizada por un aprendiz en un curso (Usuario autenticado docente)
- Visualización de progreso de curso (Usuario autenticado aprendiz)
- Realización de una actividad del curso (Usuario autenticado aprendiz)
- Envío de respuesta a actividad (Usuario autenticado aprendiz)
- Visualización de resultado de actividad (Usuario autenticado aprendiz)

5.1.3.2. Requisitos no funcionales

En la especificación de requisitos, los requisitos no funcionales tienen como objetivo detallar los criterios que pueden usarse para juzgar la operación del sistema.

Para este proyecto, los requisitos no funcionales se han organizado en 4 grandes grupos: fiabilidad, usabilidad y accesibilidad, portabilidad y seguridad. Cada requisito se detalla en el Anexo 2: Especificación de requisitos no funcionales.

5.2. Diseño

En este apartado del cuerpo del trabajo se procede a hacer una descripción del diseño de software de la aplicación y la plataforma *Adaptive Learning*, comenzando por su arquitectura, continuando con la persistencia de sus datos y finalizando con las interfaces, el flujo entre las mismas y sus estilos.

5.2.1. Arquitectura

A continuación, se va a describir la estructura, funcionamiento e interacción entre los diferentes componentes del sistema, en primer lugar desde una perspectiva lógica, relativa a los elementos de software, y a continuación desde una perspectiva física, relativa a la infraestructura tecnológica.

5.2.1.1. Arquitectura lógica

El patrón arquitectónico utilizado en la estructura de la aplicación Adaptive Learning es en esencia el **patrón MVC** (Modelo-Vista-Controlador) según el cual los datos y la lógica de negocio se separan de la interfaz de usuario y del módulo encargado de gestionar los eventos y las comunicaciones. El patrón MVC describe una estructura compuesta por 3 capas diferenciadas:

- Capa de **modelo de datos**: el modelo guarda tanto los datos de la aplicación como la lógica de negocio, y su único propósito consiste en almacenar y gestionar los datos.
- Capa de **presentación** o visualización: es el conjunto de objetos vista de una aplicación y presentan al usuario la información del modelo.
- Capa de **control**: los controladores se diseñan para que puedan responder a los distintos eventos originados por los objetos vista y gestionar el flujo de datos entre los objetos del modelo de datos y la capa de visualización.

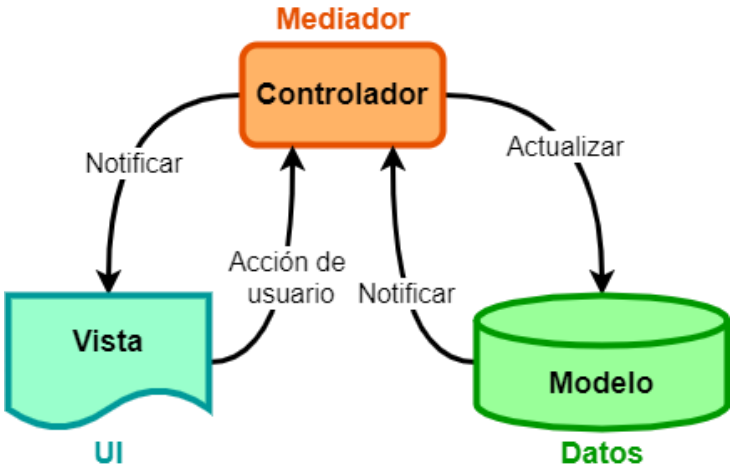

Ilustración 13: Patrón Modelo-Vista-Controlador.

Esta estructura permite crear un software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

5.2.1.2. Arquitectura física

El modelo de diseño de software de la plataforma *Adaptive Learning* se basa en la **arquitectura cliente-servidor**, en la que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. En este modelo, el servidor suele ser una máquina bastante potente con un hardware y software específico que actúa de depósito de datos y funciona como un sistema gestor de base de datos o aplicaciones. Los clientes son normalmente las estaciones de trabajo que solicitan servicios al servidor.

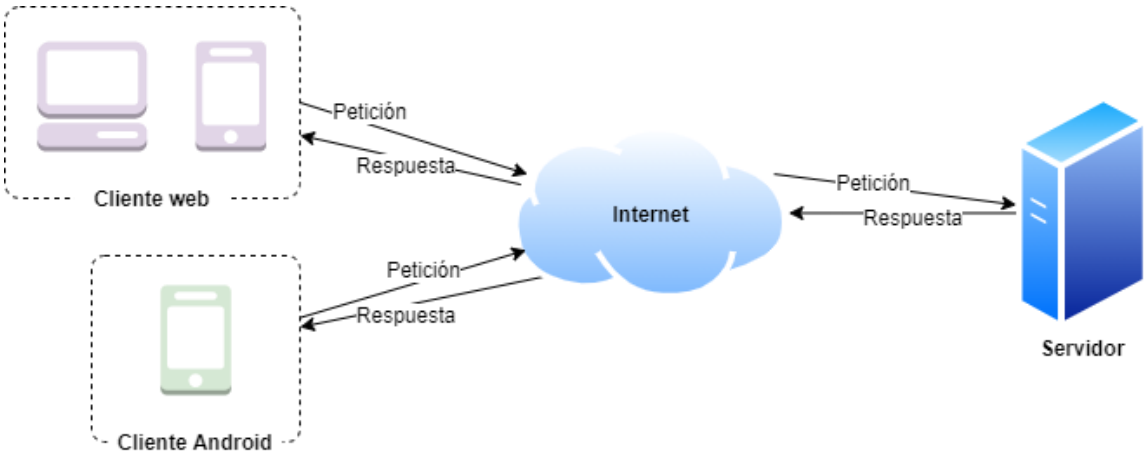

Ilustración 14: Arquitectura cliente-servidor.

La estructura modular de este modelo facilita la integración de nuevas tecnologías y el crecimiento de la infraestructura, lo que favorecerá la estabilidad de las soluciones.

En el caso de nuestro proyecto, en el lado del **servidor** hacemos uso de **Laravel**, un *framework* de código abierto que proporciona una sintaxis elegante y expresiva para generar servicios web bajo PHP. Este se comunica con una base de datos *mySQL* con el objetivo de almacenar y recuperar la información del sistema. Los clientes son los encargados de enviar las peticiones, a través de internet y mediante el protocolo HTTPS, al servidor cuando requieren de esta información o desean guardarla, haciendo uso de una API desarrollada para tales tareas.

Como **clientes**, actualmente se dispone de una **aplicación web** desarrollada en Angular a la que se puede acceder a través de un navegador desde cualquier dispositivo con conexión a internet. Además, se procede a añadir a la arquitectura, como nuevo cliente, una **aplicación Android** nativa que hará uso de la API del servidor para la obtención y guardado de los datos.

5.2.2. Diseño de la persistencia

En este apartado se especifica el diseño del sistema de persistencia con el cual se conseguirá preservar la información y los datos generados por los usuarios para poder recuperarlos y utilizarlos posteriormente.

5.2.2.1. Persistencia en servidor

Con el objetivo de disponer de los datos del usuario desde cualquier cliente del sistema se hace necesario el uso de una **base de datos MySQL** que se comunicará con el servidor para almacenar y recuperar dicha información siempre que el cliente la solicite.

Así, se ha creado una base de datos capaz de almacenar la información sobre usuarios, actividades de aprendizaje, competencias, cursos, etc. y todas las relaciones existentes entre ellos que necesita el sistema.

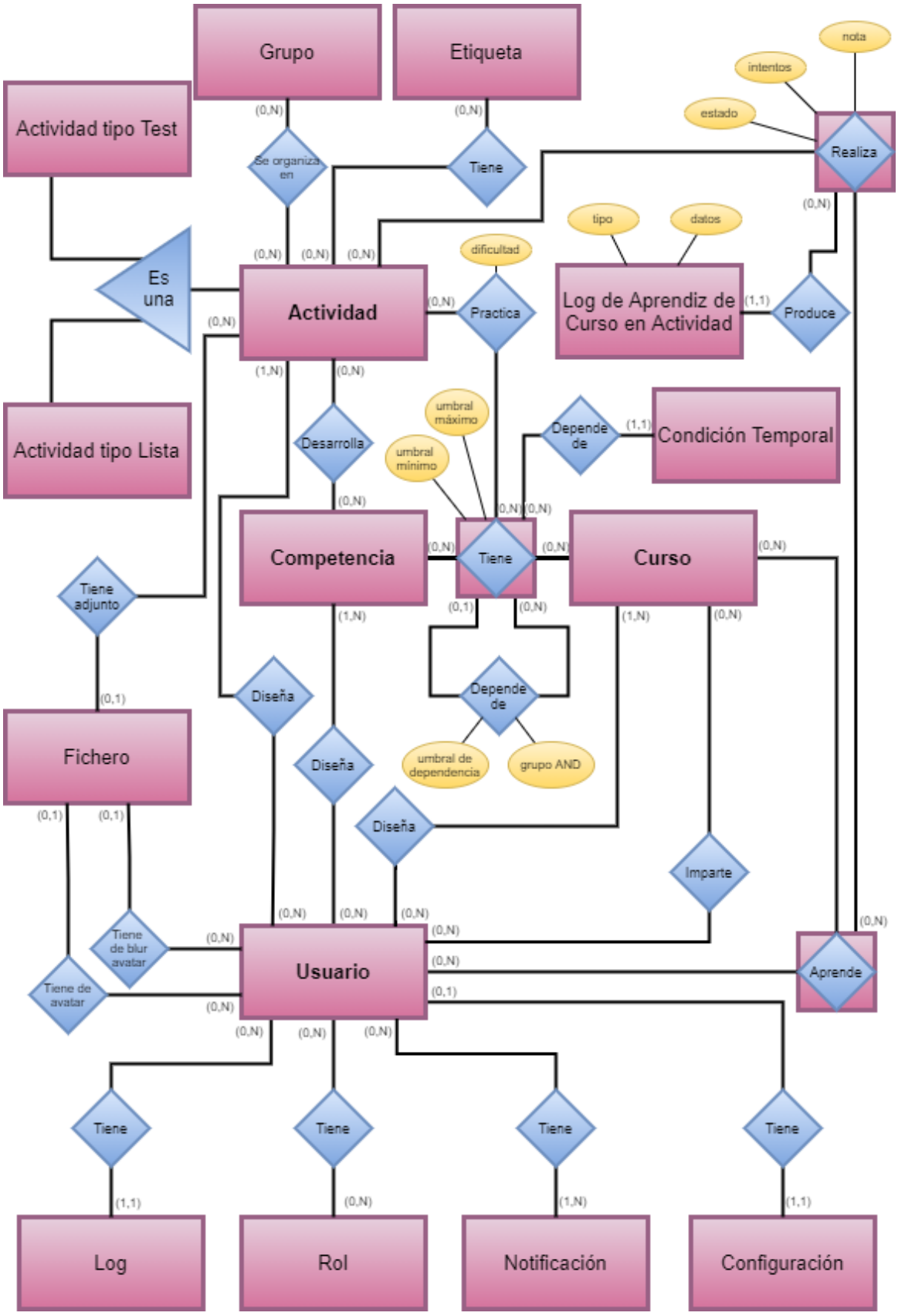

Ilustración 15: Modelo entidad-relación.

A continuación, se procede a explicar brevemente cada tabla para adquirir una idea aproximada del funcionamiento de la base de datos.

En primer lugar, se encuentran las cuatro tablas básicas **users**, **skills** (competencias), **activities** y **courses**. Estas tablas almacenan la información sobre los elementos centrales de la plataforma. En los registros de las tres últimas se guarda quién es el usuario creador de los mismos.

Seguidamente se encuentran las tablas intermedias que relacionan a los usuarios con los elementos anteriores de diferente manera: **course_designer**, **activity_designer** y **skill_designer** pretenden almacenar quiénes son los diseñadores de un curso, una actividad o una competencia respectivamente; **course_teacher** almacena los docentes de un curso y **course_learner** los aprendices de un curso.

En **skill_activity** se almacenan las actividades relacionadas con una competencia (relación que se establece al crear/editar una competencia o una actividad) debido a que el usuario la desarrollará cuando las realice.

En **course_skill** se encuentran las competencias que se han añadido a un curso y **course_skill_arc** es donde se guardan las dependencias entre competencias del curso junto al umbral de la conexión/dependencia. Esta tabla contiene un campo “and_group” que, si tiene un valor, indicará que esa dependencia forma parte de una puerta AND junto al resto de dependencias que tengan ese mismo valor en dicho campo. Por otro lado, se dispone de la tabla **course_skill_time_conditions**, que como su nombre indica, almacena condiciones temporales para una competencia de un curso de manera que el aprendiz tan solo pueda entrar a realizar actividades de la misma si las condiciones se cumplen. Estas relaciones se establecen al crear/editar un curso.

En **course_skill_activity** tenemos las actividades que se han “activado” para una determinada competencia de un curso (es decir, la relación entre **course_skill** y **activities**), y es en esta tabla donde se almacena la dificultad que se desea para la actividad. Como veremos posteriormente, la dificultad será el valor que se sumará a la puntuación del aprendiz que realice dicha actividad con éxito. Estas relaciones se establecen también al crear/editar un curso.

Dado que la realización de una actividad por un aprendiz afecta a todas las competencias de un curso en las que ésta esté activada, se hace necesario almacenar el progreso de

un aprendiz en la realización de una actividad de un curso en la tabla **course_learner_activity**, que relaciona **course_learner** con **activities**. Aquí se encuentra el campo “level”, que almacena el resultado obtenido por el aprendiz al realizar la actividad, en una nota del 0 al 10. Para saber la puntuación que suma una actividad a una competencia concreta para el aprendiz se calcula la multiplicación de la dificultad de la actividad dentro de la competencia (anteriormente mencionada) por el “level” obtenido por el aprendiz dividido entre 10. La relación entre un **course_learner** y una **activity** se establece cuando el aprendiz comienza a realizar una actividad de una competencia en un curso.

Además, como interesa saber el proceso por el que ha pasado el aprendiz a la hora de realizar la actividad, se almacena una serie de registros o logs cada vez que el usuario produce algún cambio en la misma o se entra o sale de ella. Esta información queda almacenada en **course_learner_activity_logs**. Los logs pueden ser de los siguientes tipos:

- START: indica cuándo el aprendiz comienza la actividad.
- PAUSE: indica cuándo el aprendiz ha salido de la actividad sin haberla finalizado.
- RESUME: indica cuándo el aprendiz ha vuelto a la actividad.
- LAST_CONNECTION: indica la última vez que se registró que el aprendiz estuviera en la actividad.
- CHANGE_PROGRESS: indica cuándo el aprendiz realizó cambios en la actividad. En este tipo de log se debe utilizar el campo “data” para almacenar cuáles son esos cambios. Cuando el aprendiz sale y vuelve a entrar a la actividad, si hay un CHANGE_PROGRESS para la **course_learner_activity** posterior a cualquier log de tipo SAVE y FINISH, al aprendiz se le ofrece la posibilidad de cargar dichos cambios no guardados, ya que se entiende que se marchó dejándose la actividad a medio hacer y quizá quiera recuperar su avance.
- SAVE: indica cuándo el aprendiz ha pulsado en “Guardar actividad” y debería almacenar también cuál es el estado actual de la actividad, para poder recuperarlo si el aprendiz sale y vuelve a entrar en la actividad.
- FINISH: indica cuándo el aprendiz ha pulsado en “Finalizar actividad”. Debería además almacenar el estado final de la actividad, así como la puntuación que ha obtenido en ella.

En las tablas **activity_test** y **activity_list** se almacena la información necesaria para generar la actividad de cada tipo a la hora de mostrársela al usuario, así como lo necesario para obtener la nota del aprendiz una vez la finalice.

Por último, se pueden encontrar otras tablas relacionadas con una actividad, como pueden ser los archivos adjuntos, que se almacenan en **files** y se relacionan con la actividad en **activity_file**, las etiquetas de la actividad, que se almacenan en **labels** y se relacionan con la actividad a través de **activity_label**, o los grupos de actividades que se almacenan en **clusters** y se relacionan con la actividad en **activity_cluster**. Todas estas relaciones se almacenan a la hora de crear/editar una actividad.

5.2.2.2. Persistencia en aplicación

Dado que se desea crear un sistema ubicuo, la información almacenada en el dispositivo debe ser la mínima. Sin embargo, sí se hace necesario salvar ciertos datos que faciliten el uso de la aplicación por parte del usuario. Para este almacenamiento se hace uso de las *SharedPreferences* que ofrece Android, y que permiten guardar datos básicos como preferencias de la aplicación.

El principal dato que se necesita almacenar es el valor del ***token*** del usuario, que se obtiene cuando éste inicia sesión en el sistema con éxito. Al iniciar la aplicación se comprueba si este valor existe en las preferencias, y de lo contrario se le redirigirá al formulario de inicio de sesión. Además, cada vez que se realice una petición a la API del servidor se volverá a comprobar su existencia, y de existir, se añadirá una cabecera con el nombre “Authorization” y el valor del *token*, lo que permitirá acceder a rutas restringidas de la API.

5.2.3. Diseño de interfaces

En este apartado se procede a describir la parte del diseño relativa a las interfaces, comenzando por listar los mockups de las vistas que compondrán la aplicación, para a continuación describir la navegación entre ellas y finalmente hacer una breve especificación de la guía de estilos.

5.2.3.1. Mockups

A continuación se muestran los bocetos de las diferentes vistas que encontraremos en la aplicación, que servirán como base a la hora de comenzar con la implementación.

Splash Screen

Ilustración 16: Mockup Splash Screen.

Esta vista (Ilustración 16) es la primera que aparece al iniciar la aplicación y sirve para que el usuario sepa que ésta se está arrancando mientras se aprovecha para cargar todo lo que sea necesario. En este caso, se aprovecha para comprobar si existe un *token* almacenado en preferencias y tratar de iniciar sesión con él. La vista desaparecerá automáticamente tan pronto como se haya iniciado la sesión o comprobado que no existe el *token*.

Inicio de sesión

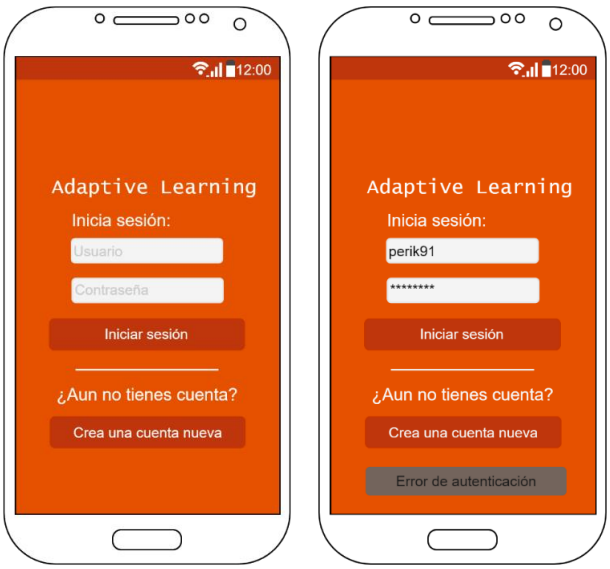

Ilustración 17: Mockups inicio de sesión.

En esta vista (Ilustración 17) encontramos el formulario de inicio de sesión, con un campo de texto para el nombre de usuario o email y otro para la contraseña. Si al pulsar en el botón “Iniciar sesión” el inicio tiene éxito, se pasará a la vista del curso del usuario. Si no tiene éxito, aparecerá, como vemos en el segundo mockup, un aviso de error de autenticación. En esta pantalla también encontramos un botón para ir al formulario de registro y crear una nueva cuenta.

Creación de cuenta

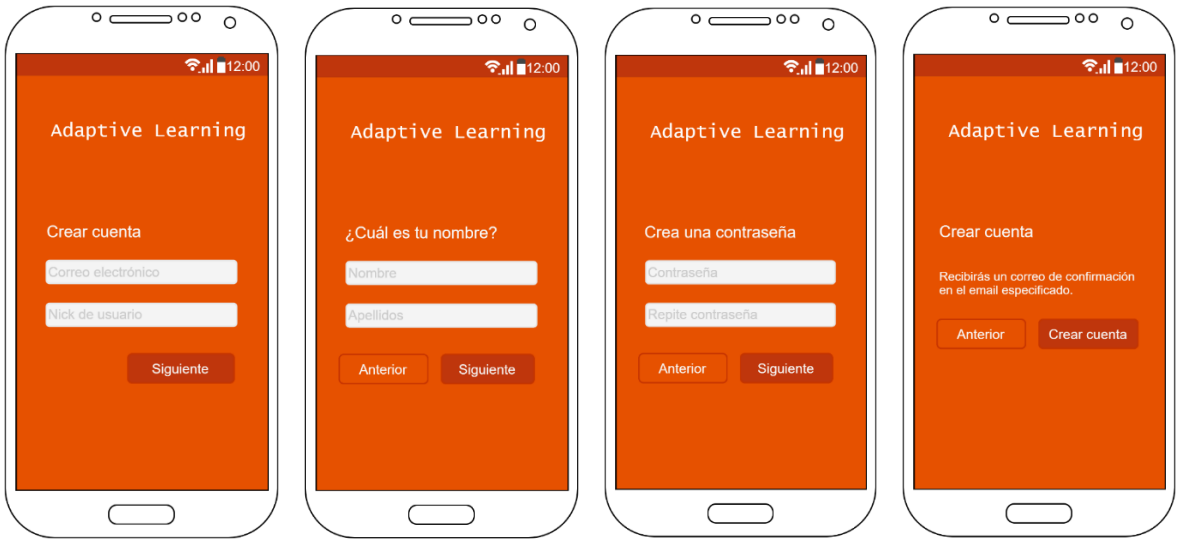

Ilustración 18: Mockups creación de cuenta.

La pantalla de creación de cuenta (Ilustración 18) será una vista paginada en la que el usuario habrá de ir completando los campos necesarios para el registro y pasando páginas hasta alcanzar el paso final en el cual se le informará de que se le enviará un email de confirmación a la dirección que haya introducido. Si en alguna de las páginas alguno de los campos no es válido, se le informará y se le impedirá pasar a la siguiente página.

Curso

Ilustración 19: Mockup curso (listado de competencias).

En esta vista (Ilustración 19) el usuario podrá visualizar su progreso en el curso que se haya cargado (por defecto, el último al que haya accedido). Se muestran aquí las diferentes competencias del curso, en forma de círculo, con la puntuación obtenida por el aprendiz en cada una de ellas. En el listado se posicionarán primero las competencias desbloqueadas sobre aquellas bloqueadas.

El título de la vista será el nombre del curso. Si el aprendiz se encuentra en más de un curso, junto al título aparecerá una flecha hacia abajo para indicar que es desplegable. En ese caso, si se hace clic en el título, aparecerá un listado de cursos disponibles (Ilustración 20). El usuario podrá elegir qué curso cargar haciendo clic en uno de ellos.

Ilustración 20: Mockup curso (selección de curso).

Si se toca uno de los círculos de la competencia se mostrará una previsualización de esta (Ilustración 21) y del progreso del aprendiz en ella. Además, se ofrecerá un botón con el que acceder a comenzar a practicar la competencia.

Ilustración 21: Mockup curso (previsualización de competencia).

Por último, si se hace clic sobre la imagen de avatar en la barra de herramientas de la aplicación se abrirá un menú (Ilustración 22) con el que acceder a las otras secciones de la aplicación: perfil, diseñador y configuración. También desde él se puede cerrar sesión. Este menú es común para todas las vistas en las que el usuario se encuentre con la sesión iniciada.

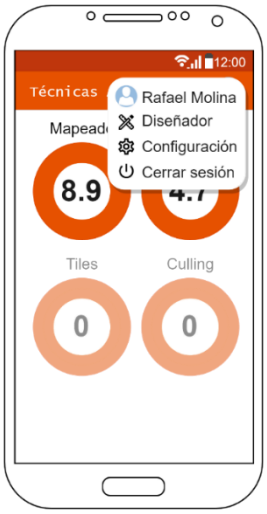

Ilustración 22: Mockup menú general

Actividad de competencia de curso

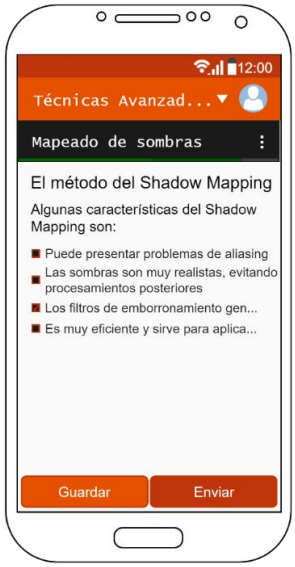

Ilustración 23: Mockup actividad de competencia de curso

Cuando el usuario accede a practicar una competencia esta es la vista que le aparecerá (Ilustración 23). Se trata de una actividad que el sistema habrá elegido para él, de aquellas que estén disponibles en la competencia que ha elegido practicar. Dependiendo del tipo de actividad aquí se visualizará un contenido diferente. En todos los casos aparecerá primero el nombre de la competencia con su progreso y debajo el título y descripción de la actividad. En la parte inferior se encontrarán dos botones: uno para que el aprendiz guarde el progreso que haya realizado en la actividad y otro para enviar su respuesta y recibir una calificación.

Al hacer clic en “Guardar”, el sistema almacenará el progreso que haya realizado el aprendiz. Este guardado se le confirmará al usuario mediante un aviso (Ilustración 24). Si sale de la aplicación o de la competencia y más tarde vuelve a acceder a practicarla, si previamente ha realizado un guardado, el progreso se cargará en la actividad y podrá continuar por donde quedó.

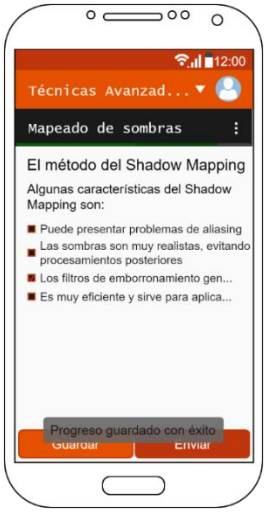

Ilustración 24: Mockup actividad de competencia de curso (progreso guardado)

Por otro lado, si se hace clic en “Enviar”, la aplicación enviará la respuesta del usuario al *backend*, que se encargará de realizar la corrección y devolver una puntuación. En ese momento se le mostrará al aprendiz la nota que ha obtenido (Ilustración 25). También se le ofrecerá un botón que si se pulsa cargará una nueva actividad de las disponibles en la competencia.

Ilustración 25: Mockup actividad de competencia de curso (respuesta enviada)

En algunos casos, tras completar una competencia se habrá superado un umbral de la competencia (el mínimo o el máximo) o el aprendiz se habrá quedado sin más actividades para realizar en ella. En esos casos, además de mostrar la puntuación, también se informará de estos eventos y se le dará la oportunidad de volver al curso pulsando en el botón “Volver” (Ilustración 26).

Ilustración 26: Mockups actividad de competencia de curso (mínimo alcanzado / máximo alcanzado / competencia agotada)

Diseñador (Competencias)

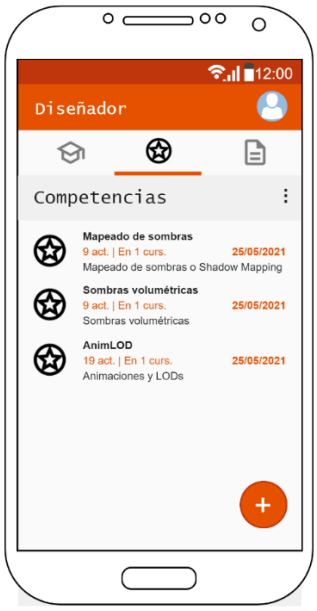

Ilustración 27: Mockup diseñador (competencias)

A esta pantalla (Ilustración 27) se accede desde el menú de la barra de herramientas de la aplicación. Se trata de una vista con pestañas con las que podemos cambiar entre los listados de “Cursos”, “Competencias” y “Actividades”. En este documento se muestran las vistas para el diseñador de competencias, pero los otros dos casos son muy semejantes: un listado con los elementos que el diseñador haya creado, un botón flotante con el que acceder a la creación de un nuevo elemento y un menú de opciones que aparece junto al título de la pestaña.

Este menú (Ilustración 28) ofrecerá las opciones de acceder a la búsqueda de elementos, activar el modo selección para tratar con múltiples elementos a la vez y acceder a la vista de creación de un nuevo elemento.

Ilustración 28: Mockup diseñador (menú de competencias)

Al activar la búsqueda, el título de la pestaña desaparecerá y se mostrará un campo de texto en el cual introducir el término por el que se desea buscar (Ilustración 29). Al pulsar el botón de búsqueda, el listado se filtrará, mostrando únicamente aquellos elementos que cumplan la condición de búsqueda.

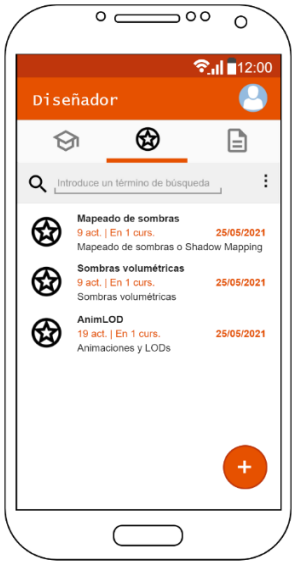

Ilustración 29: Mockup diseñador (modo búsqueda)

Por otro lado, si se activa el modo selección múltiple o si se pulsa en el icono de uno de los elementos, el título de la pestaña desaparecerá y se mostrará en su lugar un contador de elementos seleccionados, un *checkbox* para seleccionar todos los elementos y algunos botones para realizar acciones sobre los elementos seleccionados (Ilustración 30). Además, el icono de los elementos desaparecerá y aparecerá un *checkbox* que estará activado si el elemento ha sido seleccionado.

Ilustración 30: Mockup diseñador (modo selección)

Si de los botones que han aparecido se hace clic en el de la papelera, aparecerá una alerta solicitando confirmación para eliminar los elementos seleccionados (Ilustración 31). Si se selecciona “Eliminar” desaparecerán del listado los elementos que estaban seleccionados y se indicará el éxito del borrado con un aviso.

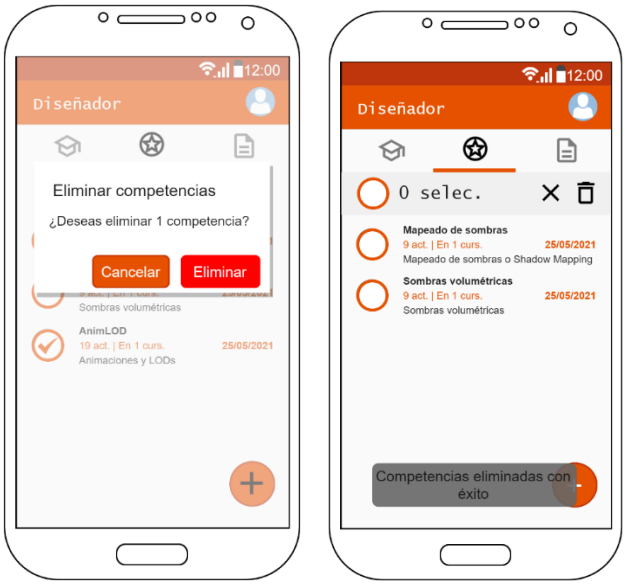

Ilustración 31: Mockups diseñador (confirmar eliminación de elemento / eliminación confirmada)

Competencia diseñada

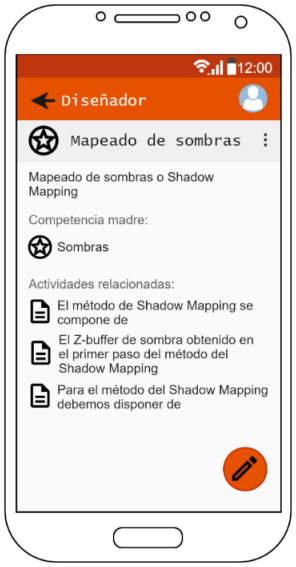

Ilustración 32: Mockup competencia diseñada

A esta vista (Ilustración 32) se accede al tocar en uno de los elementos de la lista de la vista anterior. Aquí mostramos la vista para una competencia diseñada, pero en el caso

de una actividad o un curso las vistas serán muy semejantes: el título del elemento, su descripción, información relativa al elemento y relaciones con otros elementos y un botón flotante con el que acceder a la edición de este.

Si se hace clic en el icono de menú que aparece junto al título del elemento se desplegará un menú entre los que se podrá elegir entre “Editar”, “Clonar” o “Eliminar” el elemento (Ilustración 33). Si se elige editar se abrirá la vista de edición. Si se elige clonar se abrirá la vista de creación con los campos completados con los datos del elemento.

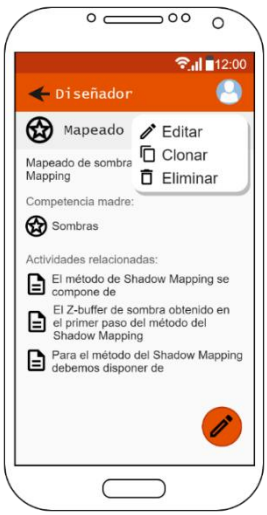

Ilustración 33: Mockup competencia diseñada (menú)

Si se elige eliminar, aparecerá una alerta que pedirá confirmación del borrado del elemento (Ilustración 34). Si se pulsa en “Eliminar” se retornará a la vista del listado tras haber eliminado el elemento del sistema.

Ilustración 34: Mockup competencia diseñada (confirmar eliminación)

Creación/edición de competencia

Ilustración 35: Mockup creación/edición de competencia

Esta vista (Ilustración 35) es la de creación o edición de una competencia, en función de si los campos están completados o no. En ella se muestran una serie de campos de texto y de selectores que permitirán al diseñador confeccionar el elemento. Además, habrá un botón flotante que permitirá guardar los cambios realizados.

Perfil

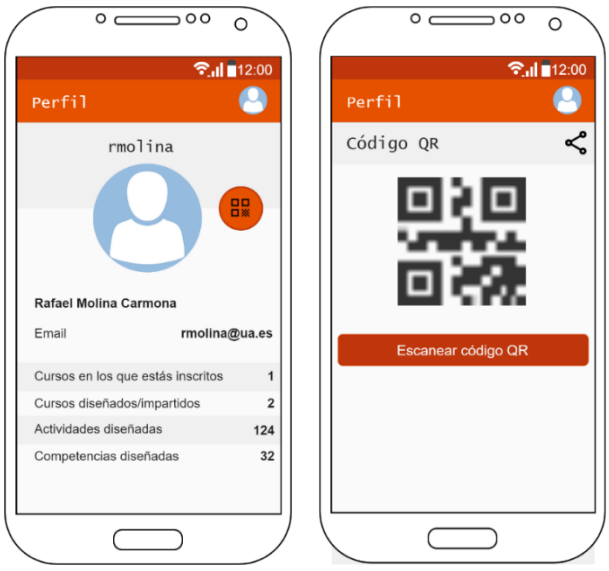

Ilustración 36: Mockups perfil de usuario y su código QR

A la vista del perfil (Ilustración 36) se accede desde el menú de la barra de herramientas, pulsando en el nombre del usuario. En ella se visualizará toda la información relativa al usuario: nombre de usuario, nombre completo, email, imagen de avatar y otros datos relacionados con el uso que haya hecho de la plataforma. Si se hace clic en el botón con el icono del código QR se mostrará el código QR del usuario, que podrá compartir externamente o ser escaneado por otro usuario.

Configuración

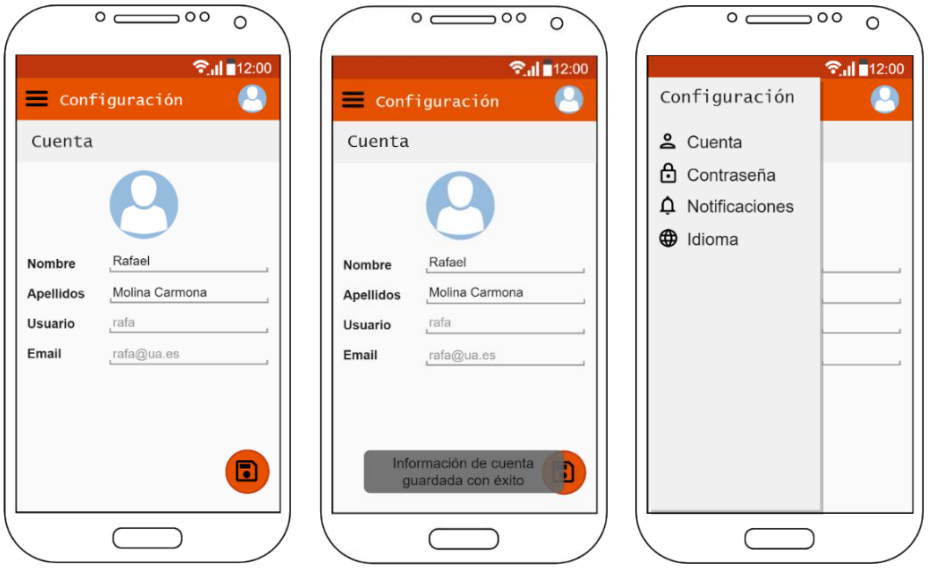

Ilustración 37: Mockups configuración (cuenta / cambios guardados / menú)

La vista de Configuración (Ilustración 37) está compuesta por varias subvistas en las que se puede cambiar diferentes aspectos de esa configuración. La que primero se carga es la de editar los datos de la cuenta: nombre, apellidos, imagen de avatar. Todas las vistas cuentan con un botón flotante para guardar los cambios, y si el guardado sucede con éxito, se muestra un aviso informando de ello. Si se hace clic en el botón del menú de la barra de herramientas se abre el menú en el que seleccionar a qué parte de la configuración se desea acceder, pudiendo cambiar a la configuración de la contraseña, las notificaciones y el idioma (Ilustración 38).

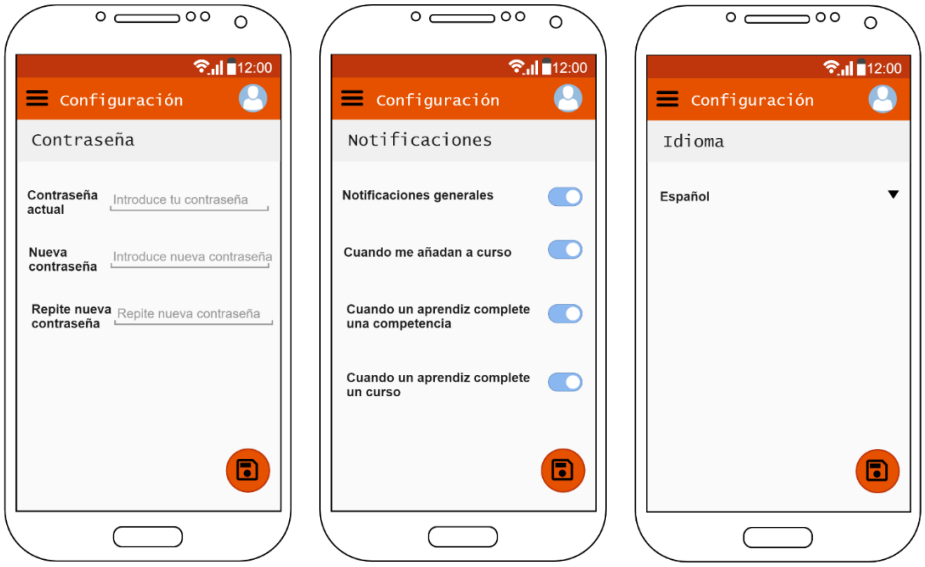

Ilustración 38: Mockups configuración (contraseña / notificaciones / idioma)

5.2.3.2. Diagramas de flujo de navegación

En este apartado de la memoria se muestran diferentes diagramas de flujo de la navegación de la aplicación que sirven para complementar la información detallada de la interfaz anteriormente proporcionada, mostrando los posibles caminos que puede seguir el usuario a la hora de utilizar la aplicación.

Para ello se ha dividido la navegación en cuatro usos diferentes que podrían hacerse de la aplicación: “inicio de sesión y creación de nueva cuenta” para usuarios no autenticados (Ilustración 39), “perfil y configuración” para cualquier usuario autenticado (Ilustración 42), “curso y realización de actividad de competencia” para los usuarios autenticados con rol de aprendiz (Ilustración 40) y “diseñador” para los usuarios autenticados con rol de diseñador (Ilustración 41).

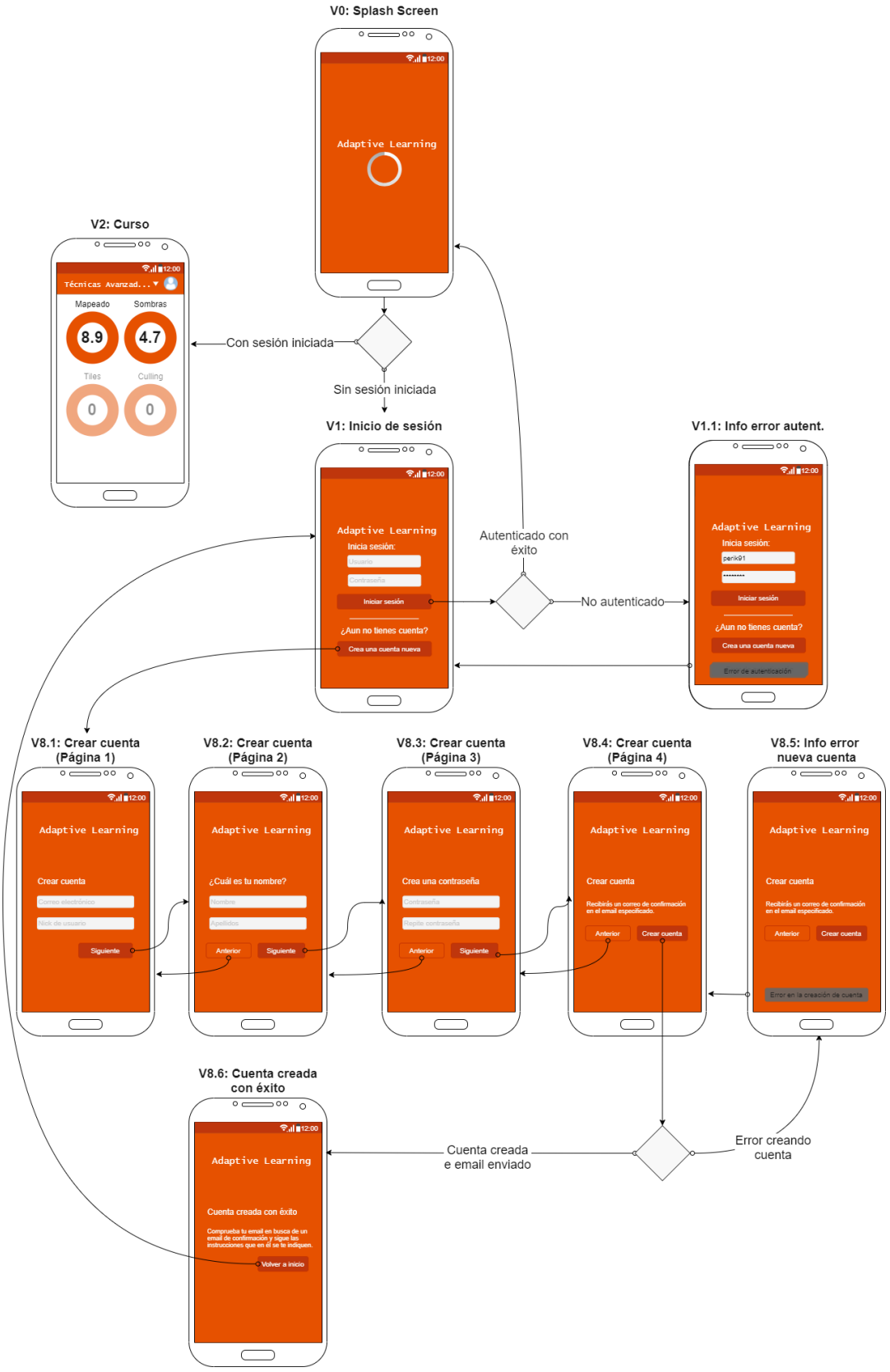

Ilustración 39: Diagrama de flujo (Inicio de sesión y creación de nueva cuenta)

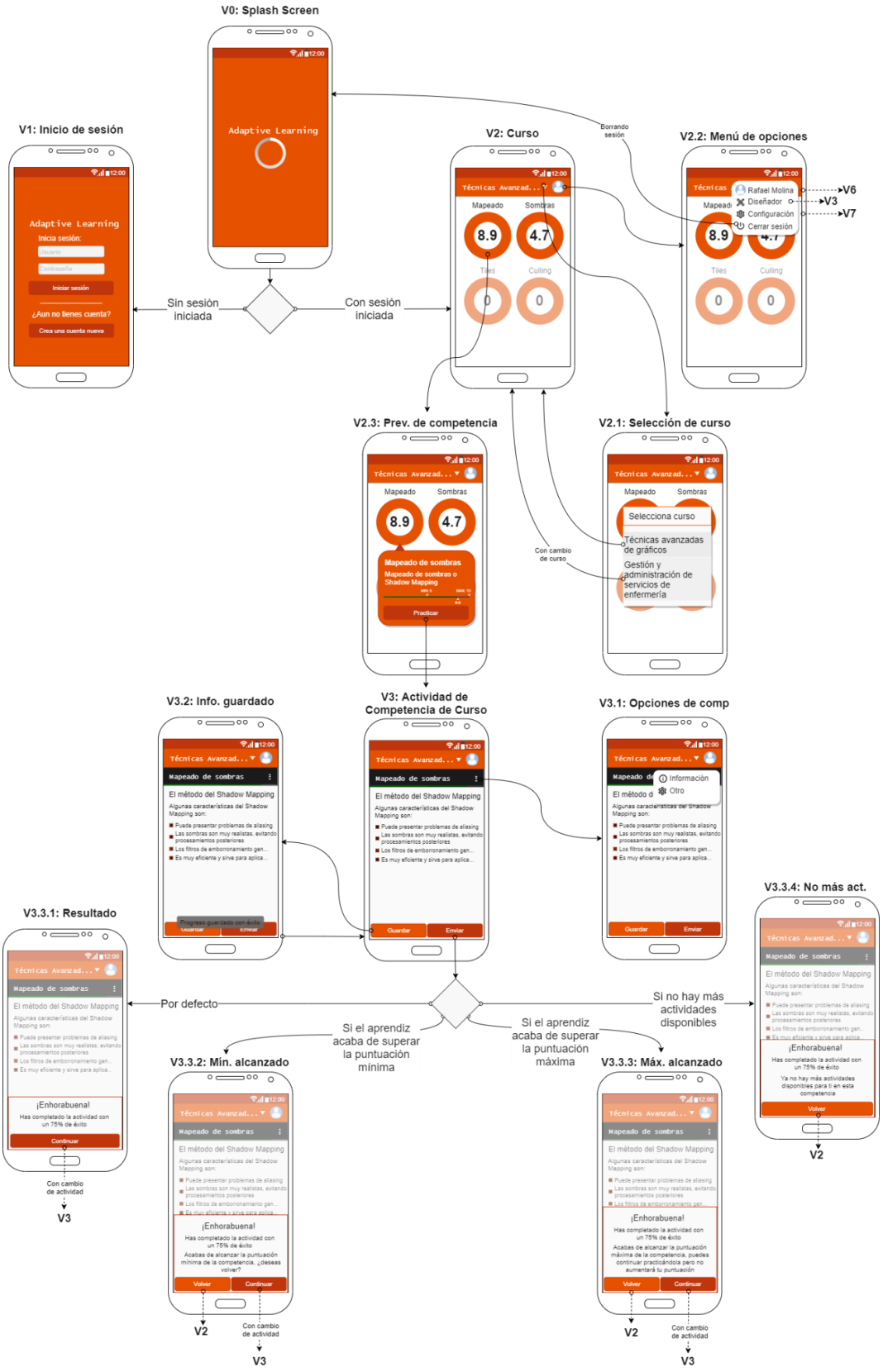

Ilustración 40: Diagrama de flujo (Curso y realización de actividad de competencia)

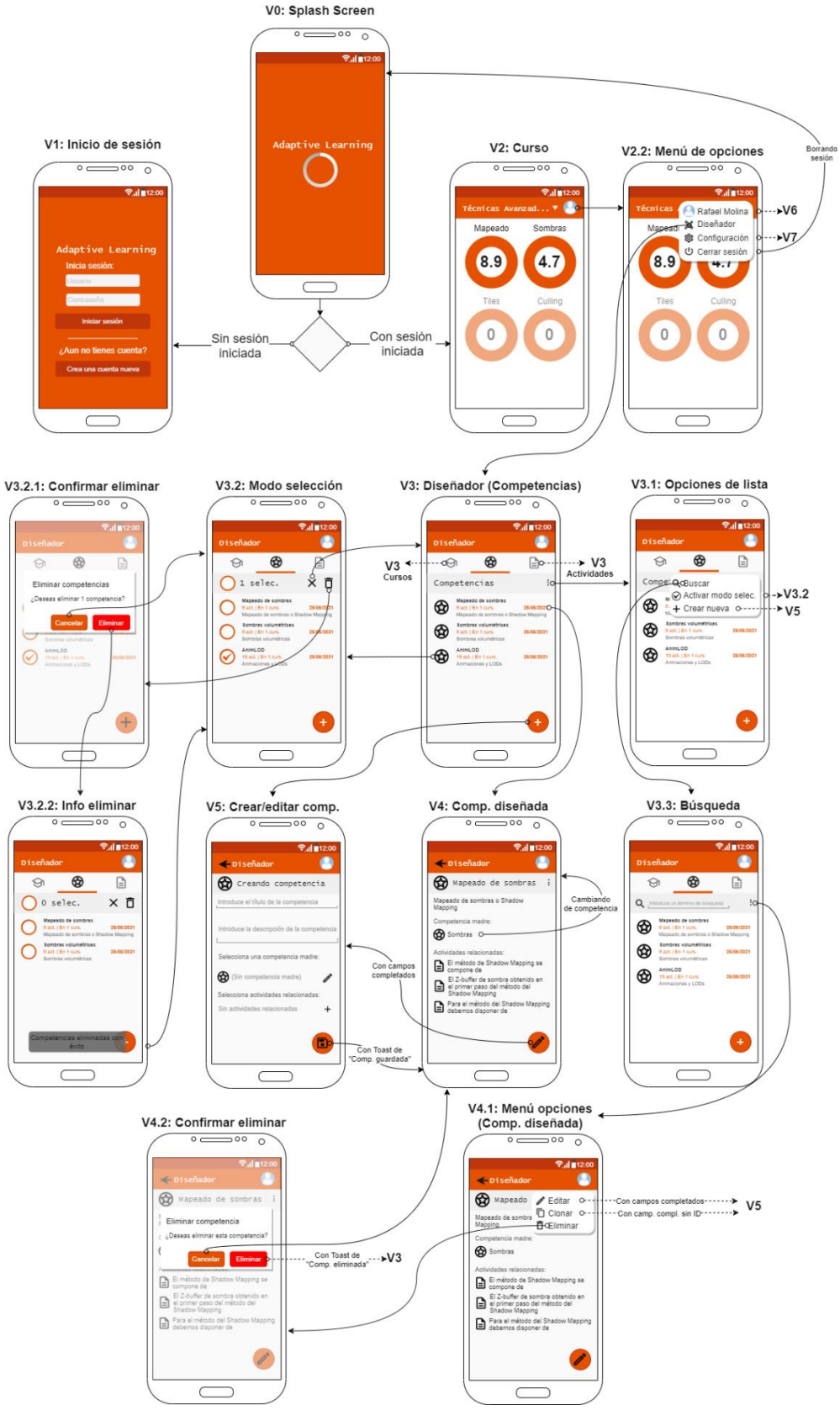

Ilustración 41: Diagrama de flujo (Diseñador de competencias)

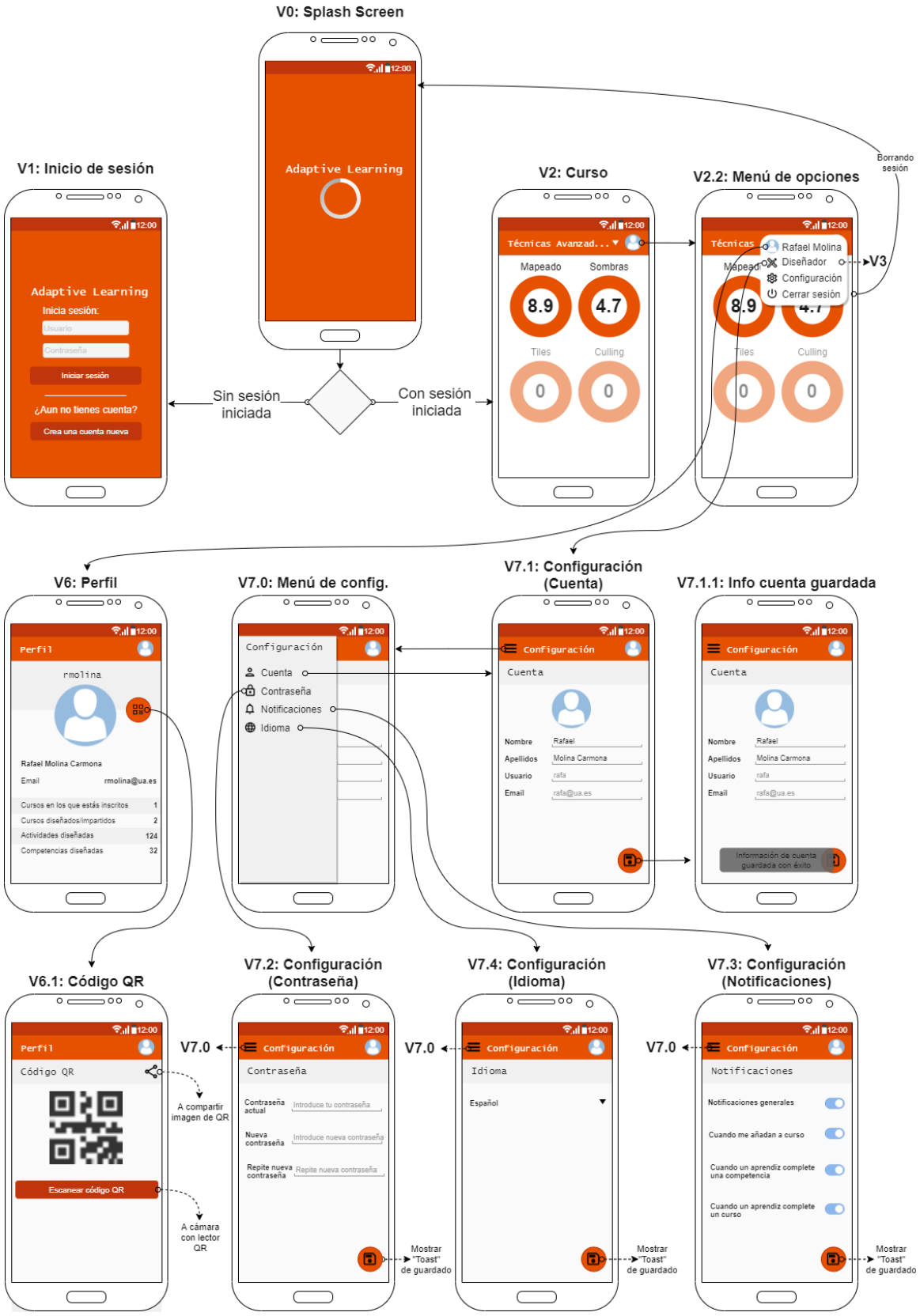

Ilustración 42: Diagrama de flujo (Perfil y configuración)

5.2.3.3. Guía de estilos

En este apartado se va a definir la guía de estilos con el objetivo de que sirva de manual de la identidad del proyecto, conteniendo los elementos necesarios para construir una imagen sólida.

Para empezar, se definen los colores de la marca (Ilustración 43). Se ha optado por una gama anaranjada para transmitir entusiasmo, exaltación y pasión. Además, su combinación con gris pretende evocar al mismo tiempo discreción y extroversión.

Ilustración 43: Colores de marca

En cuanto a la tipografía (Ilustración 44), se han escogido dos fuentes diferentes: una para el logo, desenfadada y divertida, y otra para el resto de la aplicación, más clara y sencilla para que no distraiga la atención del usuario.

Noto Sans

Ilustración 44: Tipografía de la aplicación

Por último, el logo (Ilustración 45), diseñado para la aplicación por María Virtudes Gil García y Carlos Rafael Constán Nava, consiste en un birrete al que se le han añadido tres líneas horizontales que forman parte de la imagen de marca de los logos de todos los proyectos del laboratorio UCIE Ars Innovatio. Estas tres líneas quieren hacer referencia al logo de la Universidad de Alicante.

Ilustración 45: Logo de Adaptive Learning en diferentes formatos

5.3. Implementación

Este apartado del cuerpo del trabajo contiene la documentación relativa al proceso de desarrollo del proyecto, desde la creación de este hasta la implementación final. Se proporcionan además capturas de las diferentes pantallas que forman la aplicación para que sirvan como muestra de los resultados obtenidos durante esta fase del proyecto.

5.3.1. Creación y estructura del proyecto

Para comenzar la implementación de la aplicación Android (*Android Developers*, 2021) se crea en primer lugar un nuevo proyecto en el entorno de desarrollo Android Studio (Ilustración 46). Para ello, se indica, entre otros, el nombre de la aplicación, el nombre del paquete, la ubicación de guardado y el lenguaje en el que se va a programar, en este caso, Java.

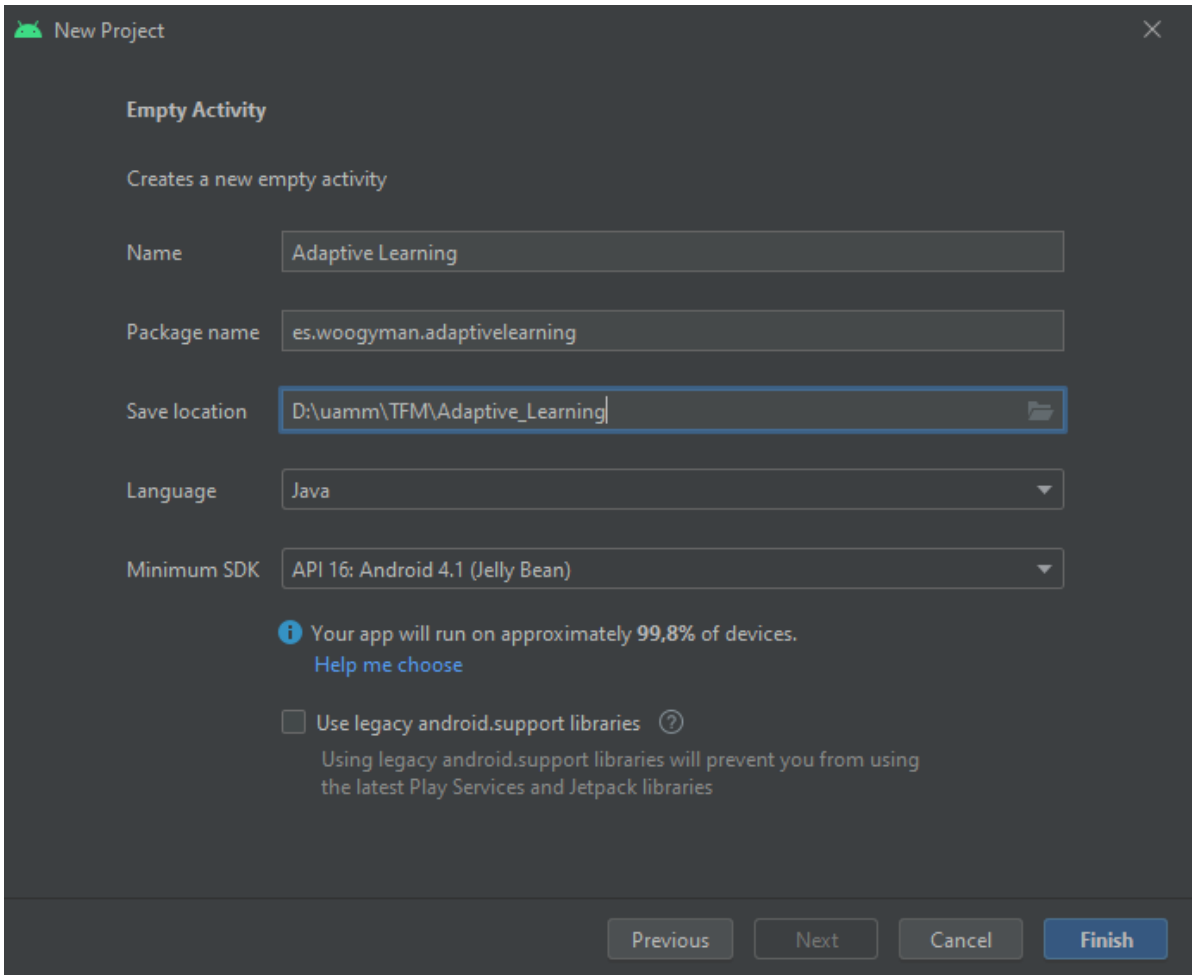

Ilustración 46: Creación de un nuevo proyecto en Android Studio

A partir de ese momento se crean los diferentes componentes, necesarios para el funcionamiento de la aplicación, almacenándolos en el proyecto según una estructura de paquetes determinada (Ilustración 47). En este caso se ha decidido seguir una estructura que combina organización por categorías con organización por funcionalidades: en los primeros niveles se organizan por tipos de componentes (interfaces, modelos, clases ayuda, servicios y clases relacionadas con la interfaz), pero dentro del paquete "`_ui._features`" se organizan por funcionalidades. En estas funcionalidades se pueden encontrar diferentes tipos de componentes, como adaptadores, *ViewModels*, actividades o fragmentos.

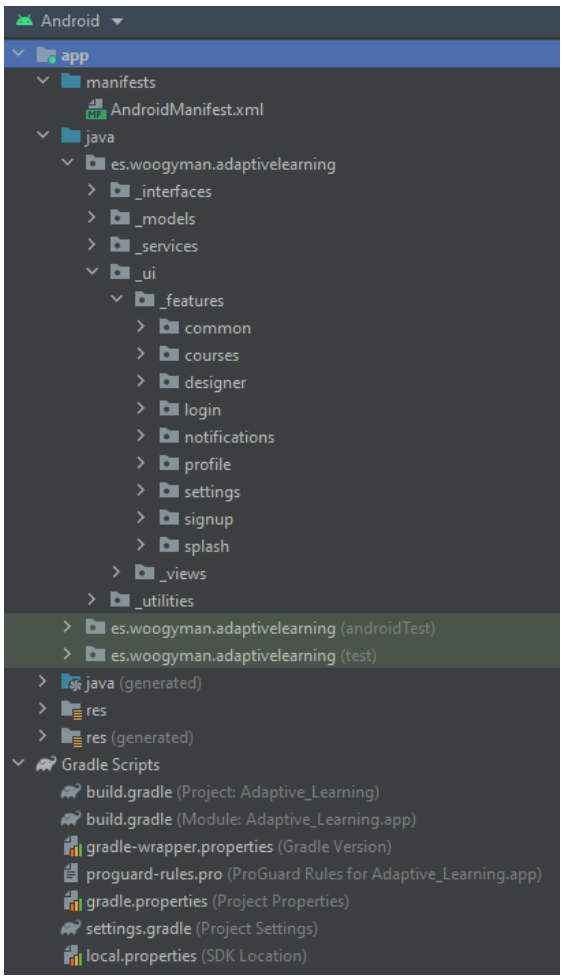

Ilustración 47: Estructura de paquetes del proyecto en Android Studio

Las **interfaces** en este caso son principalmente *callbacks* utilizados para retornar valores cuando se termina de realizar una operación o para avisar de la ocurrencia de un determinado evento.

Los **modelos** contienen la estructura de los diferentes objetos que intervienen en el sistema, como son los usuarios, los cursos, las competencias o las actividades.

Los **servicios** son componentes que pueden realizar operaciones de larga ejecución en segundo plano.

Las **clases de utilidades** o de ayuda son clases que contienen únicamente métodos estáticos y que no tienen estado ni pueden ser instanciados. Contienen un conjunto de métodos que pueden ser utilizados desde cualquier punto de la aplicación.

Las **clases de la interfaz** contienen, como hemos mencionado anteriormente, diferentes **componentes agrupados por funcionalidades**, pero también clases de tipo **vista**.

A continuación, se procede a explicar con mayor detenimiento algunos de los componentes más importantes del sistema.

5.3.2. Modelos

Se han creado modelos para los diferentes objetos que intervienen en el sistema, así como para almacenar y manejar otro tipo de información estructurada.

En primer lugar, se ha creado una clase base abstracta llamada `ElementModel` que contiene los atributos y métodos comunes de los diferentes objetos. Los atributos son por ejemplo el id del elemento, su título y su descripción. Como métodos tiene, por un lado, los *getters* y *setters* de estos atributos, pero también el método `updateFromJsonString` para extraer de una *string* con formato JSON la información de dichos atributos, u otros métodos para realizar el paso contrario, convertir el objeto en JSON o en *string*.

Los modelos que extienden esta clase son por ejemplo `ActivityModel`, `SkillModel`, `CourseModel`, `UserModel` o `NotificationModel`. Cada uno dispone de sus propios atributos y métodos, pero todos cuentan con un método `updateFromJsonString` para obtener de una *string* la información de sus atributos (que además llama al método homónimo de la clase que extienden) y sobrescriben el método `createJSONObject` para realizar el paso contrario.

Muchos de estos modelos tienen como atributos objetos de las otras clases mencionadas (Ilustración 48). Por ejemplo, un objeto `CourseModel` tiene el atributo `teachers` que puede contener un `ArrayList` de objetos de la clase `UserModel` para representar a los usuarios que son docentes del curso, y un `SkillModel` cuenta con el atributo

`activities` para almacenar objetos de la clase `ActivityModel`, que serán las actividades que los diseñadores han vinculado con dicha competencia.

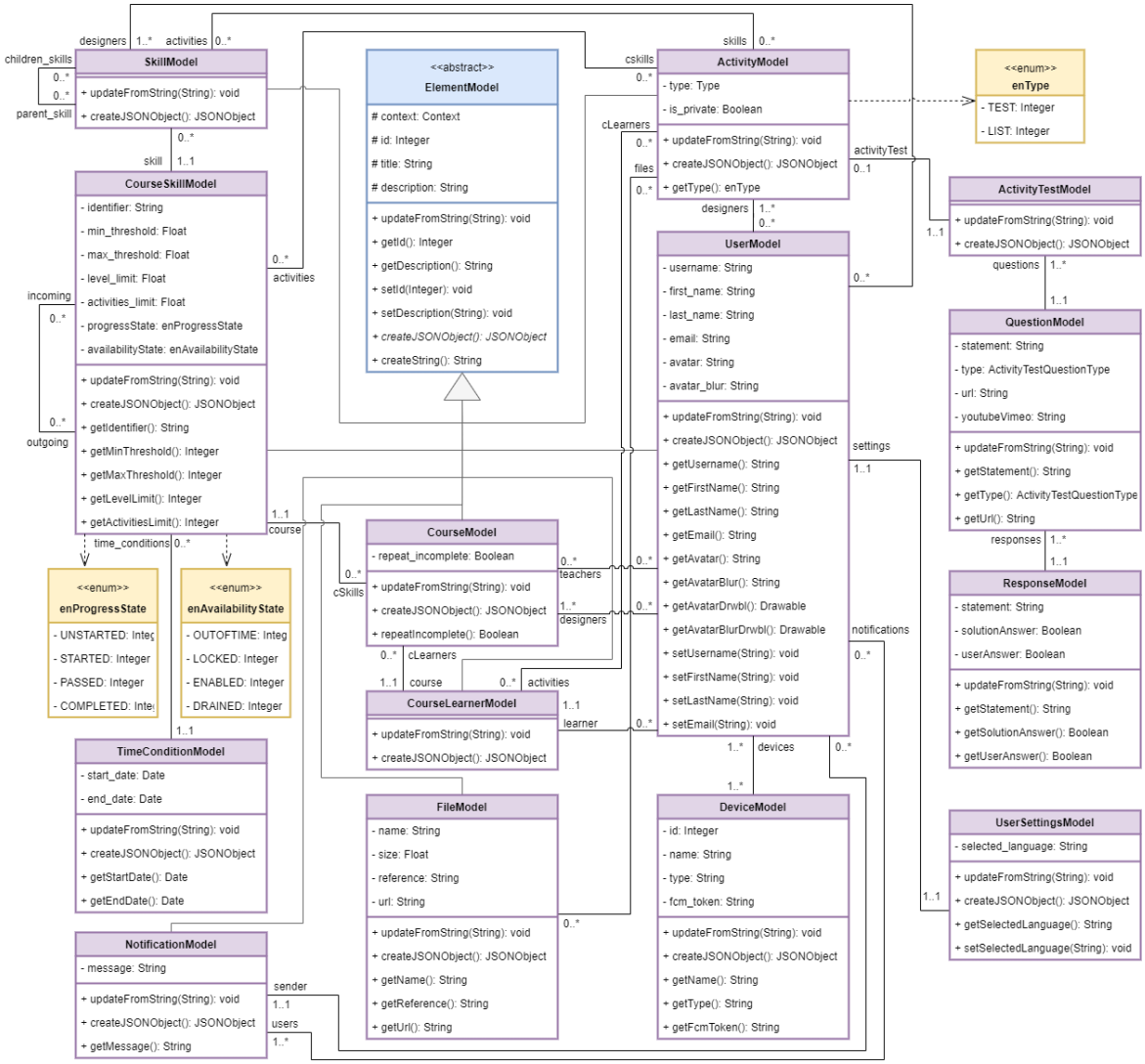

Ilustración 48: Diagrama de clases de los principales modelos de la aplicación

De los modelos mencionados, es interesante detenerse a explicar con más detenimiento la clase `UserModel`. A la hora de crear un objeto de esta clase, si se recibe un *string* para el atributo `avatar`, ésta será la *url* de la imagen de avatar del usuario. Cuando ocurre, la clase se encarga de realizar la descarga asíncrona de dicha imagen. Para ello hace uso de Picasso, una librería de Android que facilita la descarga de imágenes a partir de su *url*. Cuando se consigue descargar la imagen, ésta se guarda en un *drawable* del propio objeto y se informa a los "*listeners*" que se hayan suscrito de que la imagen ya se encuentra disponible. Cabe mencionar que, debido a la seguridad establecida en el *backend*, no se puede acceder a ningún fichero sin estar autenticado en el sistema. Es

por lo tanto necesario enviar en la petición de la imagen una cabecera con el *token* de sesión del usuario. Para ello, se ha tenido que crear un nuevo cliente HTTP para utilizar con Picasso que intercepte las peticiones y añada dicha cabecera.

El modelo `DevideModel` almacena información sobre el dispositivo, como es su nombre (marca, modelo, etc), su sistema operativo (en este caso "android" siempre) y el *token* para Firebase Messaging que permitirá al sistema enviarle notificaciones *PUSH*. Normalmente esta información se le envía al *backend* al inicio de sesión para que conozca en todo momento en qué dispositivo se encuentra el usuario.

Otro tipo de modelos creados es el de los filtros. Son modelos que se utilizan en la búsqueda de diferentes elementos para filtrar los resultados. Se envían como *string* en la petición de búsqueda y en el *backend* se interpretan y se utilizan para obtener de la base de datos únicamente los registros que interesan. Las clases de este tipo que se pueden hallar en el proyecto son por ejemplo `ActivitiesFilterModel`, `SkillsFilterModel`, `CoursesFilterModel`, `UsersFilterModel` o `NotificationsFilterModel`, y todas heredan de la clase `ElementsFilterModel`.

Por último, otro modelo importante es `HttpResponseModel` que se utiliza para extraer la información de la respuesta de una petición al *backend*. Estas respuestas se han estructurado de manera que se reciba un *string* con un mensaje de información en `message`, un array con los objetos solicitados en `data` y un array de errores en `errors` si los hubiera. Además, a partir del código de respuesta el sistema reconoce si la petición ha tenido éxito o no, y lo almacena en el atributo `success`.

Cuando la petición tiene éxito y se obtienen elementos en el array `data` es cuando se hace uso del método `updateFromJsonString` del modelo correspondiente (que depende de lo que se haya solicitado) anteriormente mencionado para convertir los *strings* recibidos en dicho array en los objetos deseados.

5.3.3. Actividades y *fragments*

A continuación, se procede a describir brevemente la implementación de las actividades y *fragments* que forman la aplicación, agrupándolos en las principales funcionalidades. La explicación completa de las mismas se puede consultar en el Anexo 3: Descripción detallada de la implementación de las pantallas de la aplicación.

Splash Screen

Ilustración 49: Vista de la Splash Screen

La *Splash Screen* (Ilustración 49) es la pantalla de transición que se muestra desde que se inicia la aplicación hasta que se han conseguido obtener los datos necesarios. Lo único que se muestra en ella es una imagen de fondo, el logo, el nombre de la aplicación y un *spinner* circular indicando que se está realizando un proceso. La encargada de manejar esta pantalla es la actividad **`SplashActivity`**, que se ha nombrado como *LAUNCHER* en el *AndroidManifest* y que por lo tanto se inicia la primera al abrir la aplicación.

Inicio de sesión

Ilustración 50: Vista del inicio de sesión

La pantalla de inicio de sesión (Ilustración 50) es aquella en la que el usuario debe introducir sus credenciales para acceder al sistema. Por esta razón en ella, además del logo y nombre de la aplicación, se muestra un formulario con los campos “Email” y “Contraseña” y un botón para “Iniciar sesión”. Además, en esta pantalla se ofrece un botón para “Crear cuenta nueva” que permite acceder al formulario de registro. La actividad que maneja esta pantalla es **`LoginActivity`**. También es esta actividad la encargada de realizar la petición de activación de cuenta tras un registro. Para ello, en el *manifest* se ha especificado con un *intent-filter* que se pueda acceder a esta actividad haciendo clic en un enlace con una *url* como la que se le enviará al usuario en el email de confirmación cuando se registre.

Ilustración 51: Vista del registro

La pantalla de registro (Ilustración 51) es un conjunto de formularios que el usuario ha de completar para enviar sus datos al sistema y así crearse una nueva cuenta. En el primer formulario ha de indicar un email y un nick de usuario, en el segundo, su nombre y apellidos, y en el último, una contraseña que habrá de repetir. Los encargados de realizar estas acciones son la actividad **`SignUpActivity`** y los *fragments*, que harán de páginas en el proceso de registro, del `SignUp1Fragment` al `SignUp5Fragment`.

Cursos

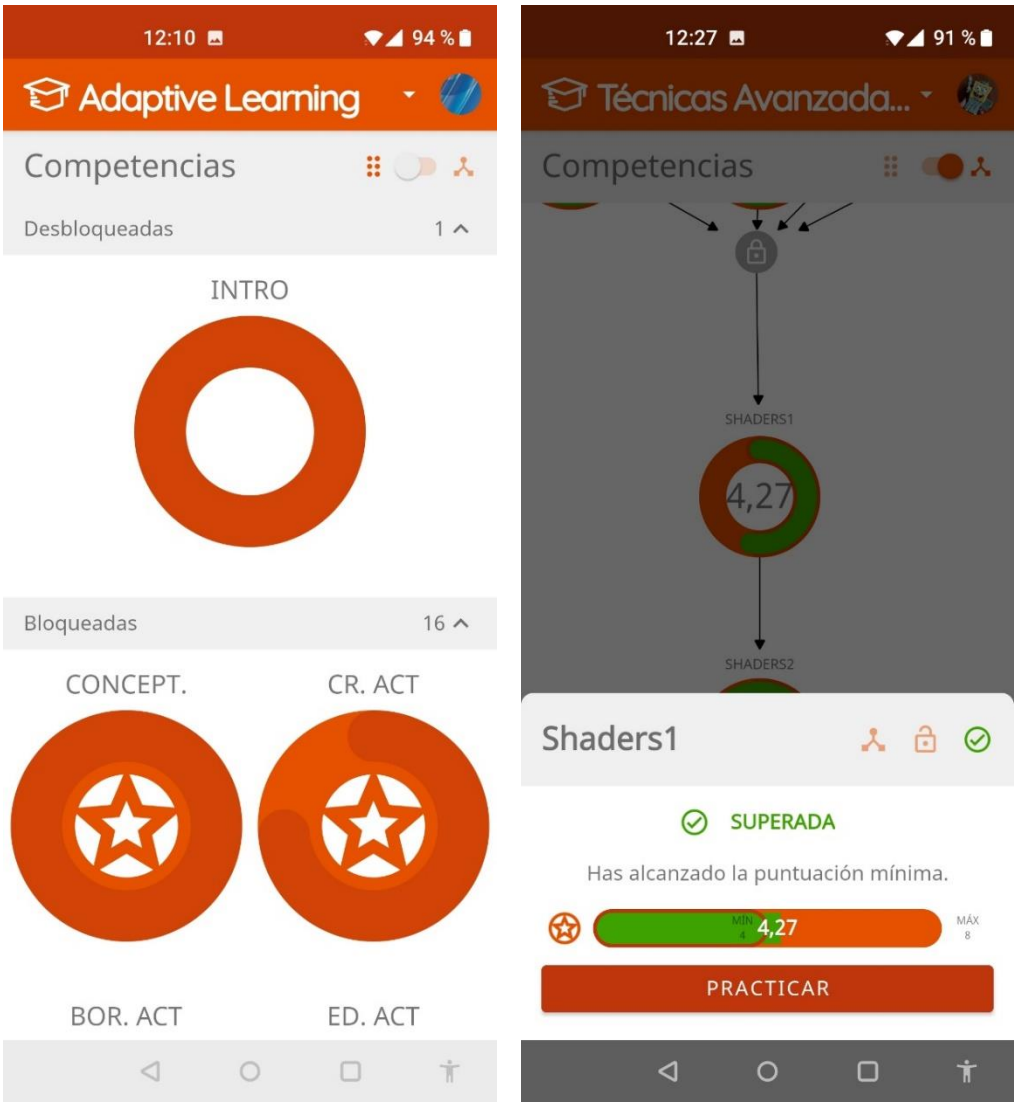

Ilustración 52: Vista de cursos (lista de competencias / previsualización de competencia)

La pantalla de los cursos (Ilustración 52) es la principal y la más compleja de todas, pues en ella sucede todo lo relativo al aprendizaje y la docencia de los cursos: la visualización del progreso en cada competencia, la realización y evaluación de actividades, etc.

La primera actividad de esta pantalla es **`CoursesActivity`**, y es la que obtiene la lista de los cursos del usuario, tanto aquellos en los que está como aprendiz como en los que está como docente. Dependiendo del curso seleccionado, realizará el cargado del *fragment* `LearningCourseFragment` o de `TeachingCourseFragment`. Estos muestran el curso en forma de una lista o de un mapa de competencias. Si se hace clic en alguna de ellas, se podrá observar el estado de esta en cuanto a disponibilidad y dependencias. También habrá un botón para acceder a practicar o evaluar la competencia.

Para practicarla, se llama a la actividad **`CourseSkillActivity`** (Ilustración 53), que es la que se ocupa de cargar las actividades de aprendizaje para que el usuario las realice en el *fragment* `CourseSkillActivityFragment`, que a su vez cargará un *fragment* u otro en función del tipo de actividad de la que se trate. Actualmente tan solo se han implementado las actividades de tipo test, por lo que el *fragment* usado es `ActivityTestFragment`. Él construye la actividad de aprendizaje cargando las preguntas que las componen en *fragments* diferentes, dependiendo del tipo de pregunta: `ActivityTestTFFragment` para las preguntas de verdadero o falso, `ActivityTestUniqueFragment` para las que tienen una única respuesta correcta y `ActivityTestMultipleFragment` para las que tienen múltiples respuestas correctas.

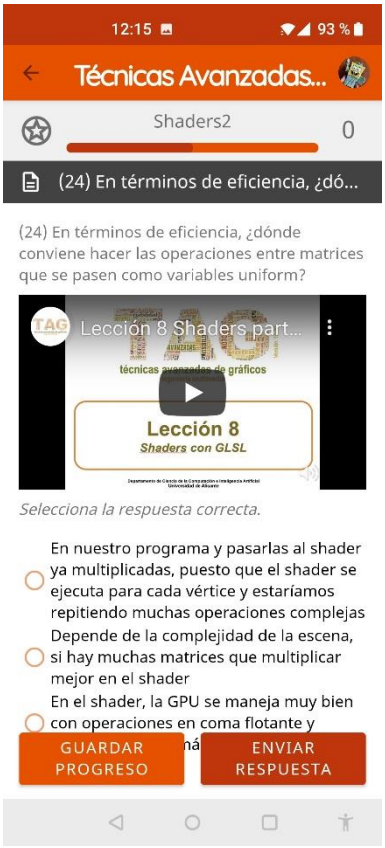

Ilustración 53: Vista de realización de una actividad de aprendizaje de una competencia

Para evaluar la competencia, se llama a la actividad **`CourseSkillTeachingActivity`**, y en ella se pueden observar todas las actividades de aprendizaje que pertenecen a la competencia seleccionada, así como su dificultad. Se puede también seleccionar un aprendiz para visualizar la nota que ha obtenido en dichas actividades y clicar en una de ellas para llamar a **`ActivityTeachingActivity`** y ver el contenido de la actividad así como las respuestas correctas de la misma junto a las dadas por el aprendiz.

Ilustración 54: Vista del diseñador (lista de competencias diseñadas)

La pantalla del diseñador (Ilustración 54) es donde sucede toda la creación de contenidos de la plataforma. Aunque por el momento tan solo se ha implementado la parte de creación/edición de competencias, también tendrá lugar aquí la creación/edición de actividades de aprendizaje y de cursos.

La actividad en la que sucede la visualización de las diferentes listas de elementos diseñados es en **`DesignerActivity`**, que hace uso del *fragment* `TabsFragment` para manejar las tres posibles pestañas diferentes. La lista de competencias se carga en el *fragment* `SkillListFragment`, y si se hace clic en un elemento, se obtiene los datos de

la competencia y se cargan en `SkillFragment` (Ilustración 55). También se puede acceder a editar una competencia o a crear una nueva, y en ambos casos se haría uso de la actividad **`SkillEditActivity`** (Ilustración 55).

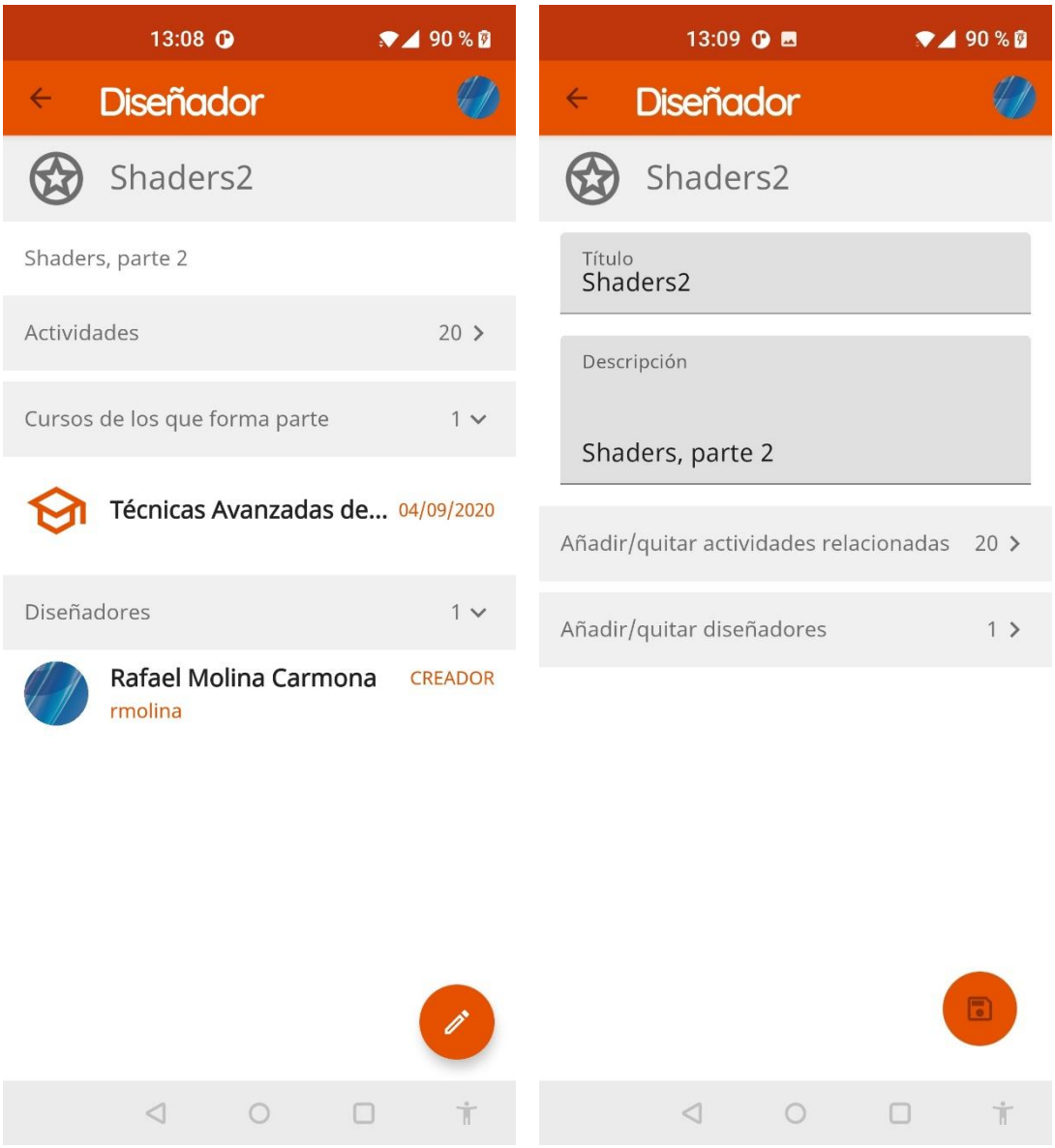

Ilustración 55: Vista de competencia diseñada / Vista de edición de competencia

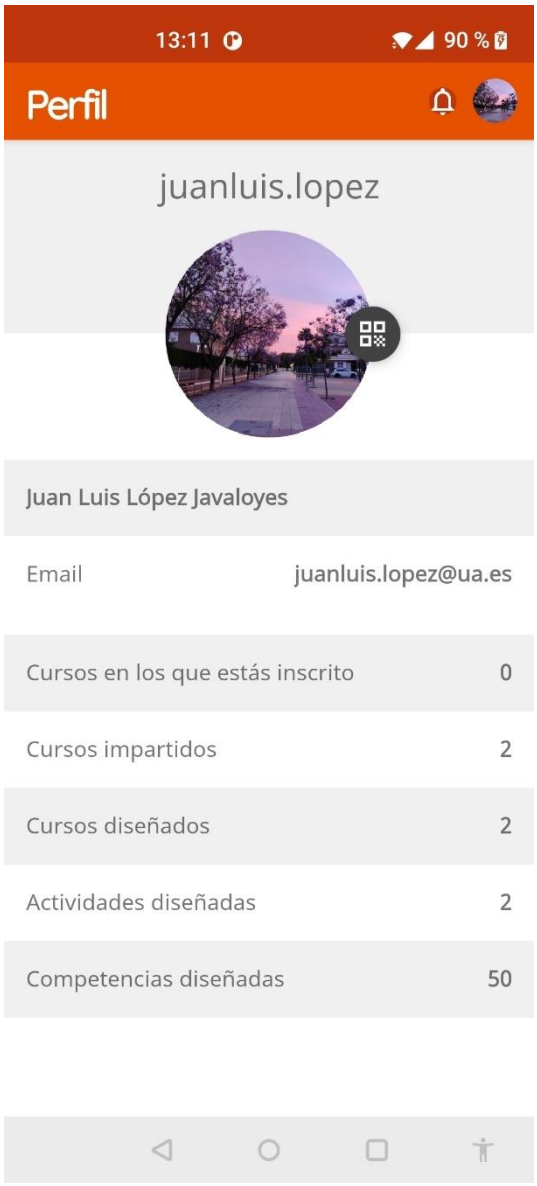

Ilustración 56: Vista del perfil

El perfil del usuario (Ilustración 56) es la pantalla en la que éste puede ver su propia información. La actividad encargada de mostrarla es **`ProfileActivity`**, que solicita la información completa del usuario enviando una petición a la API. Cuando la recibe, actualiza la interfaz con los diferentes datos, que son: su nick de usuario, su imagen de avatar, su nombre completo, su email y una serie de contadores del número de cursos en los que está inscrito, cursos que imparte, cursos que diseña, actividades y competencias que diseña. También desde aquí se puede acceder al código QR del usuario, que puede servir para añadirlo a un curso como docente o como aprendiz.

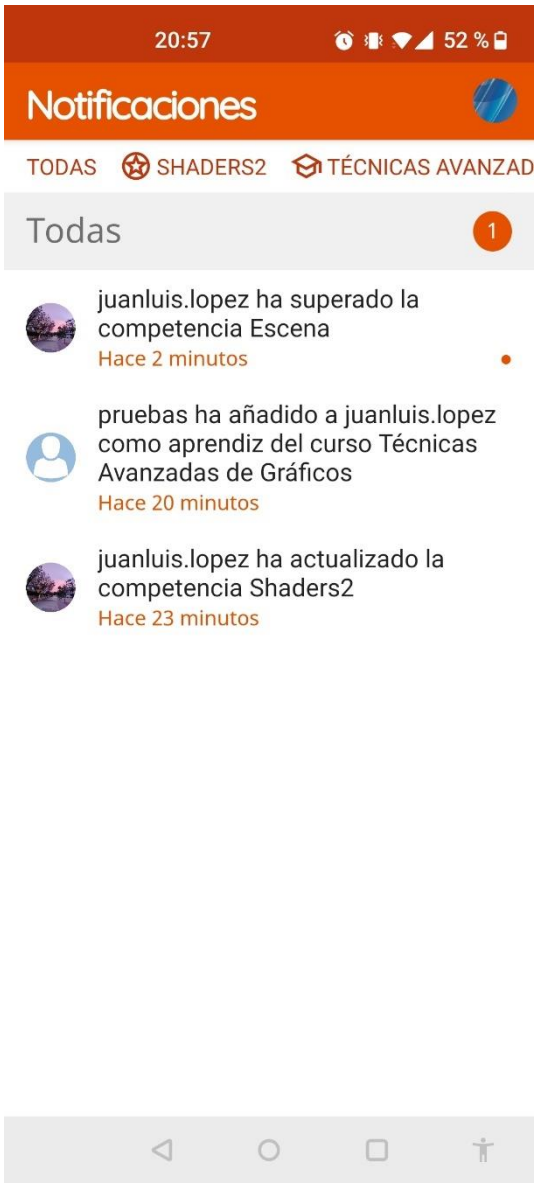

Ilustración 57: Vista de notificaciones

La pantalla de notificaciones (Ilustración 57) es donde se muestran las notificaciones del usuario. La actividad encargada de realizar esta operación es **`NotificationsActivity`**. El número de notificaciones nuevas se muestra en un contador naranja, y en el listado se distinguen de las antiguas por el punto naranja que aparece a la derecha. Se puede tocar sobre una notificación para marcarla como vista o hacer un toque largo sobre una para marcarla como no vista.

Configuración

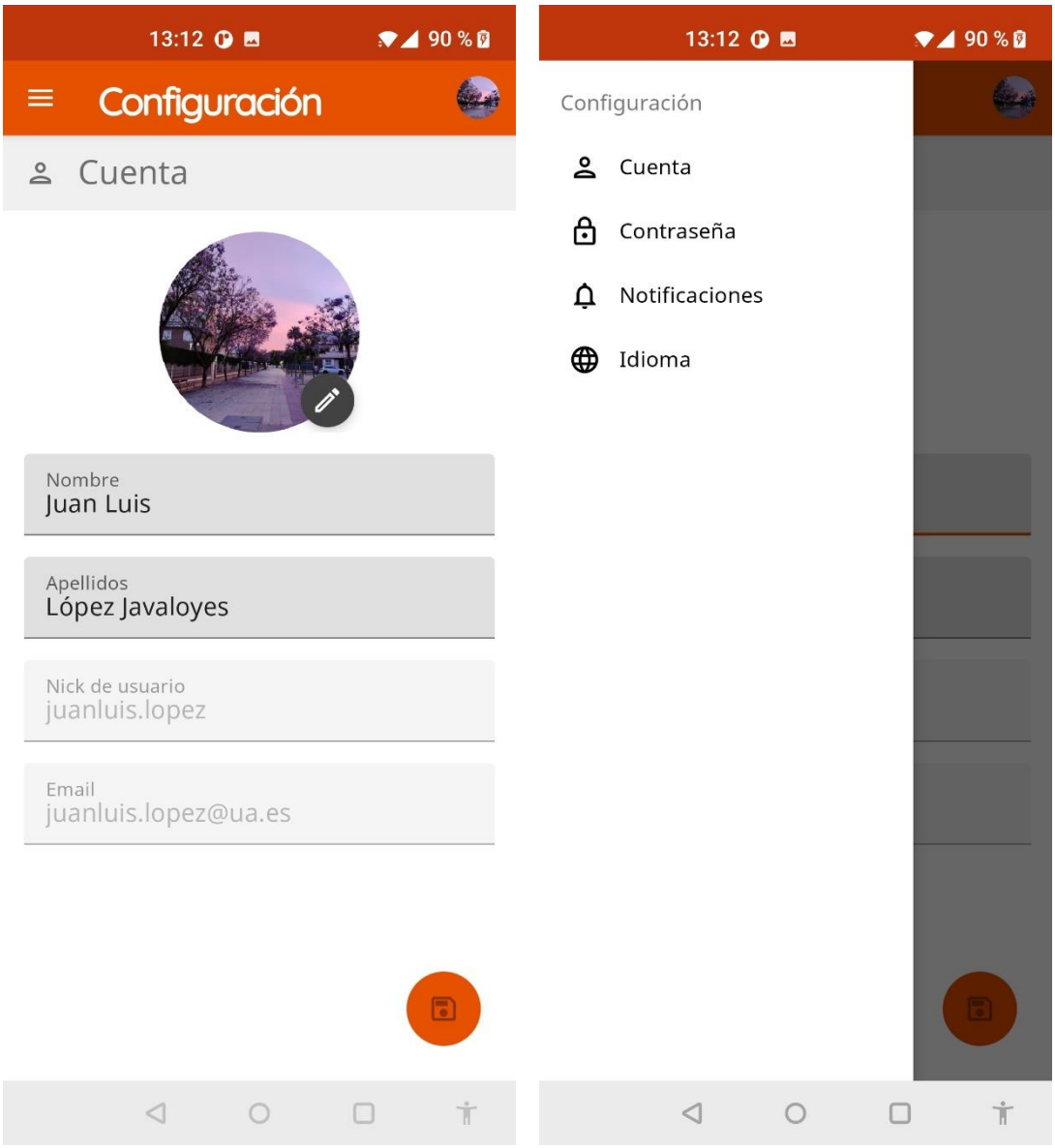

Ilustración 58: Vista de la configuración (Configuración de la cuenta / Menú)

La pantalla de configuración (Ilustración 58) es donde el usuario puede modificar las opciones de la app y sus preferencias. La actividad en la que sucede esto es **`SettingsActivity`**, que es la que carga uno u otro *fragment* dependiendo de la sección de la configuración que elija ver el usuario.

5.3.4. *Utilities* (Utilidades)

Las clases *utilities* se han empleado en esta aplicación para extraer funcionalidades que pudieran ser reutilizadas en varios puntos del código. Todos los atributos y métodos son estáticos, por lo que no es necesario instanciar ningún objeto de estas clases para utilizarlas.

A continuación se procede a explicar las utilidades más importantes de la aplicación.

HttpUtility

Esta clase contiene métodos y clases anidadas para facilitar la realización de peticiones HTTP al servidor.

La más utilizada es la clase `HTTPAsyncTask`, que hereda de `AsyncTask`, y que se encarga de realizar las peticiones a la API de *Adaptive Learning*. Para ello, su método constructor recibe los datos a enviar (en forma de objeto JSON, de *string* con formato JSON o en forma de HashMap), el tipo de petición a realizar (GET, POST, PUT o DELETE), un booleano indicando si se quiere enviar el token en una cabecera de autorización, y un *callback* para avisar de cuándo se ha recibido la respuesta a la petición. Además, al llamar al método `execute` de la clase se le pasa un *string* con la URL a la que se desea enviar la petición.

También se dispone de la clase `LoadPdfAsyncTask`, que hereda igualmente de `AsyncTask`, para recibir el `InputStream` de un archivo PDF. Esta forma de descargar dichos archivos es necesaria para poder mostrarlos en un `PDFView`, por ejemplo, en la visualización de una actividad de aprendizaje. Como en la clase anterior, al método `execute` se le pasa un *string* con la URL en la que se encuentra el archivo.

Por último, también se dispone de un método, `getPicasso`, que se emplea para construir un objeto de la clase `Picasso` utilizando un cliente HTTP personalizado con un interceptor que añadirá el *token* de autorización a la cabecera, lo que permite obtener las imágenes almacenadas en nuestro servidor protegidas con autorización.

SessionUtility

La clase `SessionUtility` contiene métodos tanto para almacenar y recuperar datos de las preferencias del sistema como para guardar temporalmente cierta información útil para el sistema en atributos estáticos.

Algunos de los métodos más importantes son `getMyUser` y `setMyUser`, utilizados para obtener y almacenar los datos del usuario que ha iniciado sesión. Esto es útil para mostrar la imagen de avatar que aparece en la barra superior de las diferentes actividades o para disponer de la id del usuario a la hora de hacer algunas peticiones.

También se dispone de métodos para obtener y almacenar la preferencia del idioma, `getSettingsSelectedLanguage` y `setSettingsSelectedLanguage`, ya que, aunque dicha preferencia se obtenga de la configuración para el usuario recibida de la base de datos, es necesario mantener esa preferencia incluso después de cerrar sesión.

De igual manera, tiene métodos para almacenar y obtener de las preferencias el último curso seleccionado por el usuario, `setLastCourseLoaded` y `getLastCourseLoadedID`, para que si sale y vuelve a entrar en la aplicación se cargue automáticamente el último curso al que accedió y no el curso por defecto que se cargaría.

Otros métodos son `logIn` y `logOut`, para almacenar y eliminar el *token* de autorización del usuario en las preferencias.

MiscUtility

La clase `MiscUtility` dispone de diversos métodos para realizar diferentes acciones, entre ellas:

- Redimensionar un *drawable* a unos determinados dp verticales y horizontales
- Codificar el archivo de una URI a Base64
- Realizar una interpolación de un valor de un determinado valor de origen a otro de destino
- Reproducir un sonido
- Obtener el MD5 de una cadena

6. CONCLUSIONES

En este apartado se realiza un repaso de los objetivos del proyecto, analizando el trabajo conseguido y describiendo posibles futuras mejoras.

6.1. Revisión de los objetivos

Una vez completada la implementación del prototipo final, se está en disposición de determinar si se ha conseguido cumplir los objetivos propuestos al comienzo del proyecto.

Se ha logrado implementar el **acceso de los aprendices a los cursos** en los que están inscritos para que puedan desarrollar las competencias que los forman. Para ello, pueden visualizar su progreso en cada curso o bien en forma de un mapa de competencias en el que pueden apreciar rápidamente las dependencias y los bloqueos entre las mismas, o bien en forma de listas de competencias agrupadas por sus estados de disponibilidad y progreso. De esta manera se consigue promover la elección del recorrido a seguir durante el aprendizaje de un curso por parte del propio aprendiz, que podrá escoger qué competencia practicar en cada momento de todas aquellas que hayan sido desbloqueadas para él.

Además, una vez que el usuario selecciona una competencia a practicar, el sistema le ofrece actividades de aprendizaje que puede realizar de manera consecutiva hasta que decida parar por haber alcanzado la puntuación máxima o hasta que ya no queden actividades para él. Estas actividades se han implementado para que puedan contar con elementos multimedia como vídeos, PDF, imágenes, etc. que favorezcan la absorción de los conocimientos antes de la realización del propio ejercicio. Por ahora, tan solo se ha implementado la posibilidad de realizar actividades de tipo test, pero en el futuro habrá una mayor variedad de tipos.

Se ha conseguido desarrollar la **evaluación y revisión del progreso de los aprendices por parte de los docentes** de un curso. Para ello, se les ha posibilitado seleccionar uno de los aprendices para que el sistema cargue el estado en el que se encuentra su curso, pudiendo, como ellos, visualizarlo en forma de lista o de mapa de competencias. Asimismo, pueden seleccionar una de las competencias para visualizar las actividades que el aprendiz ha realizado para practicarla, con su nota obtenida, su número de intentos y las respuestas que ha dado frente a las soluciones.

Para los diseñadores de contenido, se ha conseguido implementar la **visualización, creación, edición y borrado de competencias**, pudiendo relacionarlas con actividades previamente creadas, así como añadir usuarios para que sean también diseñadores. Por restricciones de tiempo, no se ha alcanzado a implementar la visualización, creación, edición y borrado de actividades de aprendizaje o de cursos, por lo que, por el momento, dichas acciones quedan limitadas a la versión web de la plataforma.

Se ha logrado implementar la **comunicación entre la aplicación y la *API REST*** alojada en el servidor de la Unidad Científica de Innovación Empresarial. Esta *API*, de implementación propia, se ha tenido que modificar y ampliar para capturar las solicitudes que le envía el cliente de la aplicación y manejarlas para retornar los datos adecuados. De esta manera, se ha conseguido que la información se encuentre disponible para el usuario independientemente del cliente que utilice.

La **integración de la aplicación con el servicio *Firebase Cloud Messaging* para el envío y recepción de notificaciones *PUSH*** se ha realizado con éxito. Estas notificaciones se han implementado para ser utilizadas como avisos para docentes del progreso de los aprendices en las competencias del curso, avisos para diseñadores de cambios realizados en competencias por parte de otros diseñadores, avisos para los usuarios de cuándo son añadidos a cursos como docentes o aprendices, etc.

Respecto al hardware, se ha logrado **mejorar la usabilidad utilizando la cámara del dispositivo**, por ejemplo, para realizar el escaneo de códigos QR y facilitar así la adición de usuarios a cursos como aprendices y como docentes. En el futuro se hará un mayor uso de estos códigos para llevar a cabo otro tipo de acciones.

En definitiva, se ha conseguido implementar un **prototipo de aplicación Android completo** con prácticamente las mismas funcionalidades que la versión web de la plataforma, e incluso añadiendo algunas que aprovechan el potencial de los dispositivos móviles. Así, se ha obtenido un producto considerablemente satisfactorio que puede ponerse en producción casi de inmediato, a falta de un par de ajustes que quedan pendientes.

Una vez finalizada la implementación se puede concluir que para la creación de esta aplicación se podría haber escogido una tecnología híbrida en lugar de Android nativo, puesto que las necesidades de rendimiento no eran excesivamente altas y no se ha

hecho uso de tecnologías con alto riesgo de incompatibilidad. Además, se podría haber reutilizado el código para otras plataformas. No obstante, ha resultado una experiencia bastante satisfactoria realizar esta implementación y muy útil para practicar el desarrollo de una aplicación completa en tecnología nativa.

6.2. Trabajo futuro

Como se ha comentado anteriormente, la limitación de tiempo ha supuesto que algunas de las funcionalidades que en el estado inicial se plantearon se hayan quedado en el tintero.

La primera de ellas sería la posibilidad de **diseñar actividades y cursos**, tal y como ahora se puede hacer con las competencias. Aunque los listados de actividades diseñadas y cursos diseñados sí que se han implementado en sus respectivas pestañas de la pantalla “Diseñador”, el borrado, edición y creación de los mismos no se ha podido desarrollar debido a la mayor complejidad que esto supone. No obstante, no se trata de una gran pérdida para el usuario, dado que estas funcionalidades se encontrarán disponibles en la versión web.

Por otro lado, queda pendiente la **ideación e implementación de actividades de aprendizaje que hagan uso de los sensores** disponibles en los dispositivos móviles. De esta manera se logrará aprovechar aún más las posibilidades de estos dispositivos para aportar jugabilidad y favorecer la utilización de la app por parte de los usuarios.

BIBLIOGRAFÍA

Android Developers. (2021). https://developer.android.com/

Codecademy. (2021). *Codecademy Go—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=com.ryzac.codecademygo&hl=es&gl=US

Coursera. (2021). *Coursera—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=org.coursera.android&hl=es&gl=US

Cross, J. (2004). An informal history of eLearning. *On the Horizon*, *12*(3), 103-110. https://doi.org/10.1108/10748120410555340

Daniel Hermawan. (2021). The Rise of E-Learning in COVID-19 Pandemic in Private University: Challenges and Opportunities. *IJORER : International Journal of Recent Educational Research*, *2*(1), 86-95. https://doi.org/10.46245/ijorer.v2i1.77

Davis, C. (2020, noviembre 13). The Future of eLearning—10 Trends To Be Aware Of. *ViewSonic Library*. https://www.viewsonic.com/library/education/10-trends-elearning-future/

Duolingo. (2021). *Duolingo—Aprende inglés y otros idiomas gratis—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=com.duolingo&hl=es&gl=US

edX. (2021). *edX - Cursos en línea de Harvard, MIT & más—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=org.edx.mobile&hl=es&gl=US

eLearning Industry. (2020, agosto 4). *The Features You Need From A Mobile Learning App*. ELearning Industry. https://elearningindustry.com/mobile-learning-app-features

EUCIM. (2021). *7 Metodologías en E-Learning que Son Tendencia*. EUCIM. https://www.eucim.es/noticias/7-metodologias-elearning-tendencia/

Khan Academy. (2021). *Khan Academy—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=org.khanacademy.android&hl=es&gl=US

LinkedIn Learning. (2021). *LinkedIn Learning—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=com.linkedin.android.learning&hl=es&gl=US

nubemia. (2015, junio 15). 5 beneficios del m-learning. *nubemia*. https://www.nubemia.com/5-beneficios-del-m-learning/

Real-Fernández, A., Molina-Carmona, R., & Llorens-Largo, F. (2017). Aprendizaje adaptativo basado en competencias y actividades—[Adaptive learning based on competences and activities]. *La innovación docente como misión del profesorado : Congreso Internacional Sobre Aprendizaje, Innovación y Competitividad*, 1-6. https://doi.org/10.26754/CINAIC.2017.000001_017

Rodenes Adam, M., Salvador Vallès, R., & Moncaleano Rodríguez, G. I. (2013). E-learning: Características y evaluación. *Ensayos de economía*, *23*(43), 143-159.

Sarrab, M., Al-Shihi, H., & Khan, A. I. (2015). An empirical analysis of Mobile learning (M-learning) awareness and acceptance in higher education. *2015 International Conference on Computing and Network Communications (CoCoNet)*, 960-963. https://doi.org/10.1109/CoCoNet.2015.7411307

tekman education. (2021, marzo 4). ¿Qué es el M-learning y qué ventajas tiene? *tekman education*. https://www.tekmaneducation.com/blog/que-es-el-m-learning/

Udemy. (2021). *Udemy—Cursos Online—Aplicaciones en Google Play*. https://play.google.com/store/apps/details?id=com.udemy.android&hl=es&gl=US

Visual Paradigm. (2021, agosto 24). *Requirement Analysis Techniques*. Visual Paradigm. https://www.visual-paradigm.com/guide/requirements-gathering/requirement-analysis-techniques/

ANEXOS

Anexo 1: Especificación de requisitos funcionales

Usuario no autenticado

FR-NA-01	**Registro**
Descripción	El sistema deberá ofrecer la opción de registrar un nuevo usuario mediante la introducción de datos en un formulario y el envío de los mismos al servidor, así como la validación de este con el envío de un correo de confirmación a la dirección indicada.
Prioridad	Alta
FR-NA-02	**Validación de usuario registrado**
Descripción	Una vez el usuario haya enviado sus datos de registro, el sistema deberá asegurarse de la pertenencia de la dirección de correo electrónico introducido por él mediante el envío de un email de confirmación. Éste contendrá un enlace que el usuario deberá seguir para confirmar su identidad.
Prioridad	Media
FR-NA-03	**Recuperación de contraseña**
Descripción	El sistema deberá posibilitar la recuperación de la contraseña del usuario en el caso de que éste la haya olvidado, ofreciéndole una interfaz en la que introducir y enviar su dirección de correo con el que se registró y al que se le enviarán instrucciones para recuperar la contraseña.
Prioridad	Baja
FR-NA-04	**Confirmación de recuperación de contraseña**
Descripción	Una vez el usuario envíe la dirección con la que se registró, el sistema le enviará un correo de confirmación que contendrá un

	enlace a través del cual podrá acceder a una interfaz en la que podrá introducir una nueva contraseña
Prioridad	Baja
FR-NA-05	**Inicio de sesión**
Descripción	El sistema deberá ofrecer la posibilidad de autenticarse en el sistema, guardando en caso de éxito el *token* de sesión de manera local para utilizarlo en posteriores peticiones que requieran autenticación.
Prioridad	Alta

Usuario autenticado

FR-A-01	**Modificación de datos de la cuenta**
Descripción	El sistema ofrecerá la posibilidad de actualizar datos del usuario, como el nombre, los apellidos o la foto de avatar.
Prioridad	Baja
FR-A-02	**Actualización de contraseña**
Descripción	El sistema deberá ofrecer un formulario de cambio de contraseña en el cual se solicite la nueva contraseña y la contraseña actual, que el sistema verificará.
Prioridad	Baja
FR-A-03	**Cambio de idioma**
Descripción	El sistema deberá estar disponible en al menos dos idiomas, español e inglés, y el usuario podrá cambiar entre ellos según desee.
Prioridad	Baja

FR-A-04	Cierre de sesión
Descripción	El sistema deberá permitir al usuario cerrar sesión en la plataforma, eliminando así el *token* del almacenamiento local y borrando los datos de sesión.
Prioridad	Alta

Usuario autenticado (Diseñador)

FR-AD-01	Listado de actividades diseñadas
Descripción	El sistema mostrará una lista de todas las actividades que haya diseñado el usuario autenticado, pudiendo filtrarlas por nombre.
Prioridad	Alta
FR-AD-02	**Diseño de actividad**
Descripción	El sistema ofrecerá un formulario de creación/edición de actividades de aprendizaje, permitiendo introducir contenido multimedia, asignar etiquetas, seleccionar grupos de actividades, indicar los vínculos con competencias o añadir diseñadores.
Prioridad	Alta
FR-AD-02-01	**Asignación de etiquetas a una actividad**
Descripción	Al diseñar una actividad, el sistema permitirá asignar etiquetas de un listado para facilitar posteriores búsquedas de dicha actividad.
Prioridad	Baja
FR-AD-02-02	**Eliminación de etiquetas a una actividad**
Descripción	Al diseñar una actividad, el sistema permitirá eliminar etiquetas de las asignadas a la misma.

Prioridad	Baja
FR-AD-02-03	**Adición de una actividad a grupos de actividades**
Descripción	Al diseñar una actividad, el sistema permitirá seleccionar en un listado de grupos de actividades a cuáles de ellos se desea que pertenezca dicha actividad, con el objetivo de facilitar la organización del usuario diseñador.
Prioridad	Baja
FR-AD-02-04	**Eliminación de actividad de grupos**
Descripción	Al diseñar una actividad, el sistema permitirá eliminar la misma de aquellos grupos de actividades a los que se haya añadido anteriormente.
Prioridad	Baja
FR-AD-02-05	**Vinculación de una actividad con competencias**
Descripción	Al diseñar una actividad, el sistema permitirá seleccionar en un listado de competencias diseñadas aquellas que se desea vincular con dicha actividad debido a que el aprendiz las desarrollará cuando la realice.
Prioridad	Alta
FR-AD-02-06	**Eliminación de competencias vinculadas con una actividad**
Descripción	Al diseñar una actividad, el sistema permitirá eliminar las competencias que se hayan vinculado previamente con esta.
Prioridad	Alta
FR-AD-02-07	**Adición de usuarios como diseñadores de una actividad**
Descripción	Al diseñar una actividad, el sistema permitirá seleccionar en un listado de usuarios diseñadores aquellos que se desea que puedan realizar cambios en el diseño de dicha actividad.

Prioridad	Baja
FR-AD-02-08	**Eliminación de usuarios diseñadores de una actividad**
Descripción	Al diseñar una actividad, el sistema permitirá eliminar aquellos usuarios diseñadores que no sean el creador de dicha actividad, para que no puedan realizar cambios en el diseño de la actividad.
Prioridad	Baja
FR-AD-03	**Eliminación de actividades diseñadas**
Descripción	El sistema permitirá seleccionar de la lista de actividades diseñadas aquellas que desea eliminar del sistema.
Prioridad	Alta
FR-AD-04	**Listado de competencias diseñadas**
Descripción	El sistema mostrará una lista de todas las competencias que haya diseñado el usuario autenticado, pudiendo filtrarlas por nombre.
Prioridad	Alta
FR-AD-05	**Diseño de competencia**
Descripción	El sistema ofrecerá un formulario de creación/edición de competencias, permitiendo indicar los vínculos con actividades de aprendizaje o añadir diseñadores.
Prioridad	Alta
FR-AD-05-01	**Vinculación de una competencia con actividades**
Descripción	Al diseñar una competencia, el sistema permitirá seleccionar en un listado de actividades de aprendizaje diseñadas aquellas que se desea vincular con dicha competencia debido a que el aprendiz la desarrollará cuando las realice.
Prioridad	Alta

FR-AD-05-02	**Eliminación de actividades vinculadas con una competencia**
Descripción	Al diseñar una competencia, el sistema permitirá eliminar las actividades que se hayan vinculado previamente con esta.
Prioridad	Alta
FR-AD-05-03	**Adición de usuarios como diseñadores de una competencia**
Descripción	Al diseñar una competencia, el sistema permitirá seleccionar en un listado de usuarios diseñadores aquellos que se desea que puedan realizar cambios en el diseño de dicha competencia.
Prioridad	Baja
FR-AD-05-04	**Eliminación de usuarios diseñadores de una competencia**
Descripción	Al diseñar una competencia, el sistema permitirá eliminar aquellos usuarios diseñadores que no sean el creador de dicha competencia, para que no puedan realizar cambios en el diseño de esta.
Prioridad	Baja
FR-AD-06	**Eliminación de competencias diseñadas**
Descripción	El sistema permitirá seleccionar de la lista de competencias diseñadas aquellas que desea eliminar del sistema.
Prioridad	Alta
FR-AD-07	**Listado de cursos diseñados**
Descripción	El sistema mostrará una lista de todos los cursos que haya diseñado el usuario autenticado, pudiendo filtrarlos por nombre.
Prioridad	Alta
FR-AD-08	**Diseño de curso**
Descripción	El sistema ofrecerá un formulario de creación/edición de cursos,

	permitiendo indicar las competencias que lo compondrán, crear dependencias entre ellas o añadir diseñadores.
Prioridad	Alta
FR-AD-08-01	**Adición de usuarios como diseñadores de un curso**
Descripción	Al diseñar un curso, el sistema permitirá seleccionar en un listado de usuarios diseñadores aquellos que se desea que puedan realizar cambios en el diseño de dicho curso.
Prioridad	Baja
FR-AD-08-02	**Eliminación de usuarios diseñadores de un curso**
Descripción	Al diseñar un curso, el sistema permitirá eliminar aquellos usuarios diseñadores que no sean el creador de dicho curso, para que no puedan realizar cambios en él.
Prioridad	Baja
FR-AD-08-03	**Adición de competencias a un curso**
Descripción	Al diseñar un curso, el sistema permitirá seleccionar en un listado de competencias diseñadas aquellas que se desea vincular con el curso debido a que se quiere que el aprendiz realice las actividades que se hayan vinculado con ellas previamente.
Prioridad	Alta
FR-AD-08-03-01	**Edición de configuración de una competencia de un curso**
Descripción	Al diseñar un curso y tras añadir una competencia, el sistema permitirá editar la configuración de esta. Se podrá modificar el identificador de la competencia, la puntuación mínima (valor de la puntuación del aprendiz en la competencia a partir del cual se considera que ha superado dicha competencia), la puntuación máxima (valor que como mucho alcanzará la puntuación del aprendiz en la competencia, aunque continúe realizando

	actividades), limitaciones del número de actividades realizables por el aprendiz, etc.
Prioridad	Alta
FR-AD-08-03-01-01	**Activación de actividades de una competencia de un curso**
Descripción	Al diseñar un curso y editar la configuración de una competencia, el sistema permitirá marcar de una lista de las actividades de aprendizaje vinculadas con dicha competencia aquellas que se desea que estén disponibles para realizar por los aprendices cuando entren a practicar la competencia.
Prioridad	Alta
FR-AD-08-03-01-02	**Editar la puntuación/dificultad de la actividad de una competencia de un curso**
Descripción	Al diseñar un curso y editar la configuración de una competencia, el sistema permitirá modificar, en la lista de las actividades de aprendizaje activadas, la dificultad de cada una de ellas, o lo que es lo mismo, los puntos que el aprendiz obtendrá una vez que realice dichas actividades con éxito.
Prioridad	Alta
FR-AD-08-03-02	**Adición de dependencias entre competencias de un curso**
Descripción	Al diseñar un curso y tras añadir dos o más competencias, el sistema permitirá establecer una dependencia entre las mismas, indicando una puntuación umbral por la cual la competencia dependiente estará disponible para el aprendiz únicamente cuando haya alcanzado dichos puntos en la competencia de origen. El valor de la puntuación umbral no debería ser mayor que el umbral máximo de la competencia de origen.
Prioridad	Alta

FR-AD-08-04	Eliminación de competencias de un curso
Descripción	Al diseñar un curso, el sistema permitirá seleccionar de las competencias añadidas previamente aquellas que se desea que ya no estén en el curso.
Prioridad	Alta
FR-AD-09	**Eliminación de cursos diseñados**
Descripción	El sistema permitirá seleccionar de la lista de cursos diseñados aquellos que desea eliminar del sistema.
Prioridad	Alta

Usuario autenticado (Docente)

FR-AT-01	Listado de cursos impartidos
Descripción	El sistema mostrará una lista de todos los cursos en los cuales el usuario se encuentra como docente.
Prioridad	Alta
FR-AT-02	**Adición de usuarios como docentes de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar de un listado de usuarios aquellos que se desea añadir como docentes de dicho curso.
Prioridad	Media
FR-AT-03	**Eliminación de docentes de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar del listado de docentes aquellos que ya no se desea que actúen con ese rol en el curso.

Prioridad	Media
FR-AT-04	**Adición de usuarios como aprendices de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar de un listado de usuarios aquellos que se desea añadir como aprendices de dicho curso.
Prioridad	Alta
FR-AT-05	**Eliminación de aprendices de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar del listado de aprendices aquellos que ya no se desea que actúen con ese rol en el curso.
Prioridad	Baja
FR-AT-04	**Envío de invitaciones a docentes de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá indicar un listado de direcciones de correo, junto a un mensaje de invitación, a las que se enviará un email con un enlace a través del cual los usuarios podrán registrarse y unirse a dicho curso como docentes.
Prioridad	Media
FR-AT-05	**Eliminación de invitaciones a docentes de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar del listado de invitaciones a docentes aquellas que ya no se desea. Si la invitación eliminada todavía no ha sido aceptada, el usuario que recibió la invitación no podrá unirse al curso, aunque siga el enlace que se le envió.
Prioridad	Media
FR-AT-06	**Envío de invitaciones a aprendices de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá indicar

	un listado de direcciones de correo, junto a un mensaje de invitación, a las que se enviará un email con un enlace a través del cual los usuarios podrán registrarse y unirse a dicho curso como aprendices.
Prioridad	Media
FR-AT-07	**Eliminación de invitaciones a aprendices de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá seleccionar del listado de invitaciones a aprendices aquellas que ya no se desea. Si la invitación eliminada todavía no ha sido aceptada, el usuario que recibió la invitación no podrá unirse al curso, aunque siga el enlace que se le envió.
Prioridad	Media
FR-AT-08	**Visualización del progreso de un aprendiz en un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá visualizar el progreso de los aprendices: actividades realizadas, notas obtenidas, puntuaciones en cada competencia, estado del mapa de competencias, etc.
Prioridad	Alta
FR-AT-08-01	**Visualización de actividad realizada por un aprendiz en un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá visualizar las actividades realizadas por uno de los aprendices, pudiendo así ver la respuesta que éste envió para cada actividad frente a la solución de esta.
Prioridad	Alta
FR-AT-09	**Borrado del progreso de un aprendiz en la actividad de un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá borrar los puntos obtenidos y las respuestas que un aprendiz ha dado a una actividad.

Prioridad	Media
FR-AT-10	**Borrado del progreso de un aprendiz en un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá borrar los puntos obtenidos y las respuestas de todas las actividades de un aprendiz.
Prioridad	Media
FR-AT-11	**Borrado del progreso de todos los aprendices en un curso**
Descripción	En un curso del que el usuario es docente, el sistema permitirá borrar los puntos obtenidos y las respuestas de todas las actividades de todos los aprendices.
Prioridad	Media
FR-AT-12	**Salida de un curso impartido**
Descripción	En un curso del que el usuario es docente, el sistema le permitirá salir de él para que no aparezca en el listado de cursos impartidos.
Prioridad	Baja

Usuario autenticado (Aprendiz)

FR-AS-01	**Listado de cursos estudiados**
Descripción	El sistema mostrará una lista de todos los cursos en los cuales el usuario se encuentra como aprendiz.
Prioridad	Alta
FR-AS-02	**Visualización de progreso de curso**
Descripción	En un curso del que el usuario es aprendiz, el sistema ofrecerá la visualización del progreso de éste en él. De esta manera mostrará el estado del mapa de competencias para él: la puntuación que ha

	obtenido en cada competencia, el estado de bloqueo o desbloqueo de cada una de ellas, etc.
Prioridad	Alta
FR-AS-03	**Realización de una actividad del curso**
Descripción	En un curso del que el usuario es aprendiz y una vez elija una competencia para practicar, el sistema hará uso del motor de selección para escoger una actividad de aquellas disponibles para el aprendiz y se la ofrecerá para que la lea, comprenda y conteste.
Prioridad	Alta
FR-AS-03-01	**Guardado automático de cambios en actividad**
Descripción	Cuando el aprendiz esté realizando una actividad, el sistema deberá almacenar cada cambio que realice en las respuestas. Así, si el aprendiz sale de la actividad sin terminarla, la próxima vez que entre a realizarla se le ofrecerá cargar los cambios realizados anteriormente.
Prioridad	Alta
FR-AS-03-02	**Cargado de cambios guardados automáticamente en actividad**
Descripción	Cuando el aprendiz entre a realizar una actividad en la que anteriormente haya hecho algún cambio, el sistema le ofrecerá la posibilidad de cargar dichos cambios.
Prioridad	Alta
FR-AS-03-03	**Guardado de progreso en actividad**
Descripción	Cuando el aprendiz esté realizando una actividad, el sistema le ofrecerá la posibilidad de guardar el progreso que haya realizado. Así, la próxima vez que entre a realizarla se le cargará para poder continuar por donde la había dejado.

Prioridad	Alta
FR-AS-03-04	**Cargado de progreso guardado en actividad**
Descripción	Cuando el aprendiz entre a realizar una actividad en la que anteriormente haya hecho algún guardado de progreso, el sistema cargará automáticamente dicho progreso para que continúe por donde la había dejado.
Prioridad	Alta
FR-AS-03-05	**Envío de respuesta a actividad**
Descripción	Cuando el aprendiz esté realizando una actividad, el sistema le ofrecerá la posibilidad de enviar la respuesta que haya dado para que se calcule y devuelva la nota que haya obtenido.
Prioridad	Alta
FR-AS-03-06	**Visualización de resultado de actividad**
Descripción	Cuando el aprendiz haya enviado su respuesta a una actividad, el sistema le mostrará la nota que ha obtenido (de 0 a 10), los puntos que ha obtenido (que dependen de la dificultad/puntuación que el diseñador haya otorgado a la actividad) y mostrará el progreso actualizado del aprendiz en la competencia que está practicando.
Prioridad	Alta
FR-AS-04	**Salida de un curso estudiado**
Descripción	En un curso del que el usuario es aprendiz, el sistema permitirá salir de él para que no aparezca en el listado de cursos estudiados.
Prioridad	Media

Anexo 2: Especificación de requisitos no funcionales

Fiabilidad

NFR-01	**Respaldo de los datos**
Descripción	El sistema debe soportar respaldo online del progreso y los datos del usuario para poder acceder a ellos siempre que se necesiten.
Prioridad	Alta
NFR-02	**Ubicuidad de los datos**
Descripción	El sistema debe asegurar que se disponga del progreso del usuario y de sus datos tanto en la versión móvil como en la versión web de la plataforma.
Prioridad	Alta

Usabilidad y accesibilidad

NFR-03	**Distancia de 5 clics**
Descripción	El sistema debe asegurar que con un máximo de 5 clics el usuario pueda acceder a cualquier contenido de la aplicación que desee.
Prioridad	Alta
NFR-04	**Interfaz intuitiva**
Descripción	El sistema debe disponer de una interfaz sencilla e intuitiva que facilite el rápido aprendizaje de su uso por parte de todo tipo de usuarios.
Prioridad	Alta
NFR-05	**Cambio de idioma**
Descripción	El sistema debe permitir el cambio de idioma de la interfaz,

	ofreciendo al menos dos idiomas: español e inglés.
Prioridad	Alta
NFR-06	**Gamificación**
Descripción	El sistema ha de contar con elementos de gamificación, como puntuaciones, desbloqueo de contenidos, etc. con el objetivo de mejorar el atractivo de la aplicación.
Prioridad	Alta

Portabilidad

NFR-07	**Transferencia de elementos personalizables**
Descripción	El sistema debe asegurar que la configuración y las preferencias del usuario se mantengan tras realizar posibles futuras actualizaciones de versiones.
Prioridad	Alta
NFR-08	**Escalabilidad**
Descripción	El sistema habrá de mantener su óptimo funcionamiento y rendimiento tras posibles cambios o crecimiento que pueda aplicársele a lo largo de su ciclo de vida.
Prioridad	Alta

Seguridad

NFR-09	**Invalidación de sesión**
Descripción	El sistema se asegurará de invalidar la sesión de usuario cuando este cierre sesión o cuando pase un determinado tiempo.

Prioridad	Alta
NFR-10	**Almacenamiento de contraseñas**
Descripción	El sistema hará uso de una función *HASH* (que requiera un factor de trabajo lo suficientemente alto para evitar un ataque de fuerza bruta) para codificar la contraseña del usuario antes de almacenarla en la base de datos.
Prioridad	Alta
NFR-11	**Complejidad de contraseñas**
Descripción	El sistema solicitará una contraseña de relativa complejidad a la hora de registrarse o efectuar un cambio de contraseña, que deberá componerse de al menos de 8 caracteres entre los que habrá de incluir al menos una letra minúscula, una letra mayúscula y un número.
Prioridad	Alta
NFR-12	**Seguridad en las comunicaciones**
Descripción	El sistema hará uso del protocolo seguro *HTTPS* para realizar cualquier comunicación con la *API* u otros servicios.
Prioridad	Alta

Anexo 3: Descripción detallada de la implementación de las pantallas de la aplicación

Splash Screen

Ilustración 59: A3 - Vista de Splash Screen

La *Splash Screen* (Ilustración 59) es la pantalla de transición que se muestra desde que se inicia la aplicación hasta que se han conseguido obtener los datos necesarios. Lo único que se muestra en ella es una imagen de fondo, el logo, el nombre de la aplicación y un *spinner* circular indicando que se está realizando un proceso. La encargada de manejar esta pantalla es la actividad **`SplashActivity`**, que se ha nombrado como

LAUNCHER en el *AndroidManifest* y que por lo tanto se inicia la primera al abrir la aplicación.

Esta actividad lo que hace es en primer lugar obtener el *token* del dispositivo para el servicio de FireMessaging, que le servirá posteriormente al sistema para poder enviar notificaciones *Push* al dispositivo actual, si hiciera falta. Con él se crea un objeto `DevideModel` que se guarda en la sesión utilizando la clase utilidad `SessionUtility`. Seguidamente se comprueba si hay algún usuario con la sesión iniciada comprobando si existe un *token* de sesión en memoria. Si no existe, se inicia la actividad de inicio de sesión `LoginActivity`, pero si existe, se intenta iniciar sesión utilizando el *token*, de manera que si no se consigue realizar con éxito porque este no es correcto (ha caducado, ha cambiado, etc.), se inicia la actividad de inicio de sesión antes mencionada.

Si se consigue iniciar sesión con el *token* se extrae de los datos devueltos en formato JSON la información sobre el usuario y se crea un objeto de la clase `UserModel` que se almacena en la `SessionUtility` como “mi usuario”. Posteriormente, por defecto, se inicia la actividad principal, que en este caso es la de los cursos, `CoursesActivity`, aunque si se ha llegado a la *splash screen* por haber hecho clic en una notificación *Push*, se obtiene del *intent* el id de la notificación y se inicia en su lugar la actividad `NotificationsActivity`, pasándole dicho id.

Inicio de sesión

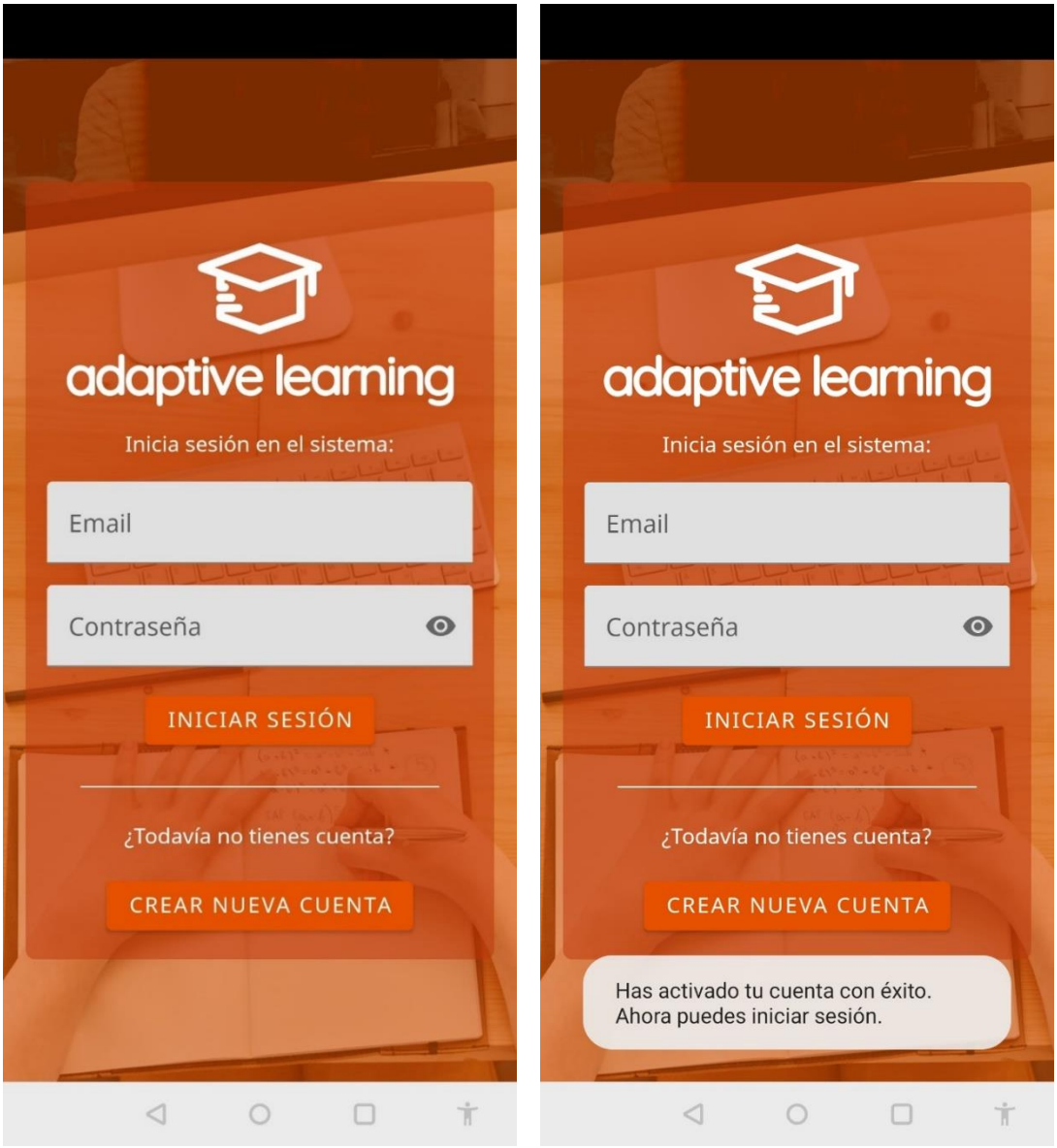

Ilustración 60: A3 - Vista de login / Aviso de cuenta activada con éxito

La pantalla de inicio de sesión (Ilustración 60) es aquella en la que el usuario debe introducir sus credenciales para acceder al sistema. Es por ello que en ella, además del logo y nombre de la aplicación, se muestra un formulario con los campos “Email” y “Contraseña” y un botón para “Iniciar sesión”. Además, en esta pantalla se ofrece un botón para “Crear cuenta nueva” que permite acceder al formulario de registro. La actividad que maneja esta pantalla es **`LoginActivity`**. También es esta actividad la encargada de realizar la petición de activación de cuenta tras un registro. Para ello, en el *manifest* se ha especificado con un *intent-filter* que se pueda acceder a esta actividad

haciendo clic en un enlace con una url como la que se le enviará al usuario en el email de confirmación cuando se registre.

Al comenzar la actividad ésta comprueba si se ha iniciado la actividad tras haber hecho clic en dicho enlace de activación enviado al usuario en el email de confirmación de registro. De ser así, se realiza la petición de activación al *backend*. Si tiene éxito, se le muestra al usuario un aviso de que se ha conseguido activar su cuenta y que puede iniciar sesión (Ilustración 60). Si no tiene éxito se le avisa de que se ha producido un error.

Cuando el usuario hace clic en el botón “Iniciar sesión” se validan los campos. Si el email o la contraseña están vacíos, se le indica con un texto bajo el correspondiente campo. Si se ha escrito en ambos, se realiza la petición de inicio de sesión. Para ello se crea un JSON con el email y la contraseña, así como la información del dispositivo que anteriormente se almacenó con el objeto de la clase `DeviceModel` en la clase utilidad `SessionUtility`.

Si la petición no tiene éxito, se le avisa al usuario del error producido. Si ha tenido éxito, se extrae de los datos recibidos el *token* de inicio de sesión y se almacena en las preferencias del sistema. Después se inicia de nuevo la actividad `SplashActivity`, que ahora, al haber un *token* de sesión almacenado, iniciará sesión con el mismo y pasará a la actividad de los cursos.

Por último, si el usuario hace clic en el botón “Crear nueva cuenta” se iniciará la actividad de registro, `SignUpActivity`.

Registro

Ilustración 61: A3 - Vista del registro (página 1 - email y nick)

La pantalla de registro (Ilustración 61) es un conjunto de formularios que el usuario ha de completar para enviar sus datos al sistema y así crearse una nueva cuenta. En el primer formulario ha de indicar un email y un nick de usuario, en el segundo, su nombre y apellidos, y en el último, una contraseña que habrá de repetir. Los encargados de realizar estas acciones son la actividad **`SignUpActivity`** y los *fragments* del `SignUp1Fragment` al `SignUp5Fragment`.

La actividad `SignUpActivity` realiza al iniciarse la carga del primer *fragment*, `SignUp1Fragment`. Al crear cualquiera de los *fragments* se le pasa un *callback* para detectar cuando el *fragment* desea que la actividad cambie de *fragment*, de manera que

si se recibe de él un número mayor de 0 es que se desea pasar al siguiente *fragment*, si no, se quiere volver al anterior. La actividad carga entonces el *fragment* correspondiente.

El primer *fragment* `SignUp1Fragment` (Ilustración 61) mostrará un formulario para introducir el email y el nick de usuario, y un botón “Siguiente” que si se pulsa verificará los campos (el email debe tener una estructura de dirección de correo electrónico mientras que el nick ha de tener al menos 3 caracteres y debe contener solo letras y números, así como puntos, barras bajas y guiones que no se encuentren al principio o al final) y le indicará a la actividad mediante al *callback* que se desea pasar al siguiente *fragment*.

En el siguiente, `SignUp2Fragment` (Ilustración 62), se ha de introducir el nombre y los apellidos, que han de tener al menos 3 caracteres, y en el siguiente, `SignUp3Fragment` (Ilustración 62), se ha de introducir una contraseña con al menos 8 caracteres y con al menos un número, una letra mayúscula y una minúscula.

Ilustración 62: A3 - Vista del registro (página 2 – nombre / página 3 – contraseña)

En el *fragment* `SignUp4Fragment` (Ilustración 63) se le indica al usuario que al pulsar en "Crear cuenta" se le enviará un correo de confirmación a la dirección que haya introducido. Si esto ocurre, se envía una petición de registro al *backend* con toda la información del usuario en forma de JSON. Si la petición no tiene éxito, se le indica mediante un aviso, pero si tiene éxito se pasa al siguiente *fragment*.

Ilustración 63: A3 - Vista del registro (página 4 – aviso / página 5 – confirmación)

En el último *fragment* `SignUp5Fragment` (Ilustración 63) se le indica al usuario que la cuenta se ha creado con éxito y que ha de comprobar su bandeja de correo en busca de un email de confirmación. Para facilitarle esta tarea se le ofrece un botón, "Abril email". Al pulsarlo la aplicación crea un *chooser* para que pueda decidir qué app de mensajería de las disponibles en su dispositivo desea abrir y así poder acceder rápidamente al email que se le ha enviado. Si todo ha ido bien, allí encontrará el correo de confirmación con el enlace de verificación (Ilustración 64). Si hace clic en dicho enlace, su dispositivo le

debería dar la opción de abrirlo con la app de *Adaptive Learning* debido al *intent-filter* que añadimos para la actividad `LoginActivity`, tal y como se ha mencionado anteriormente.

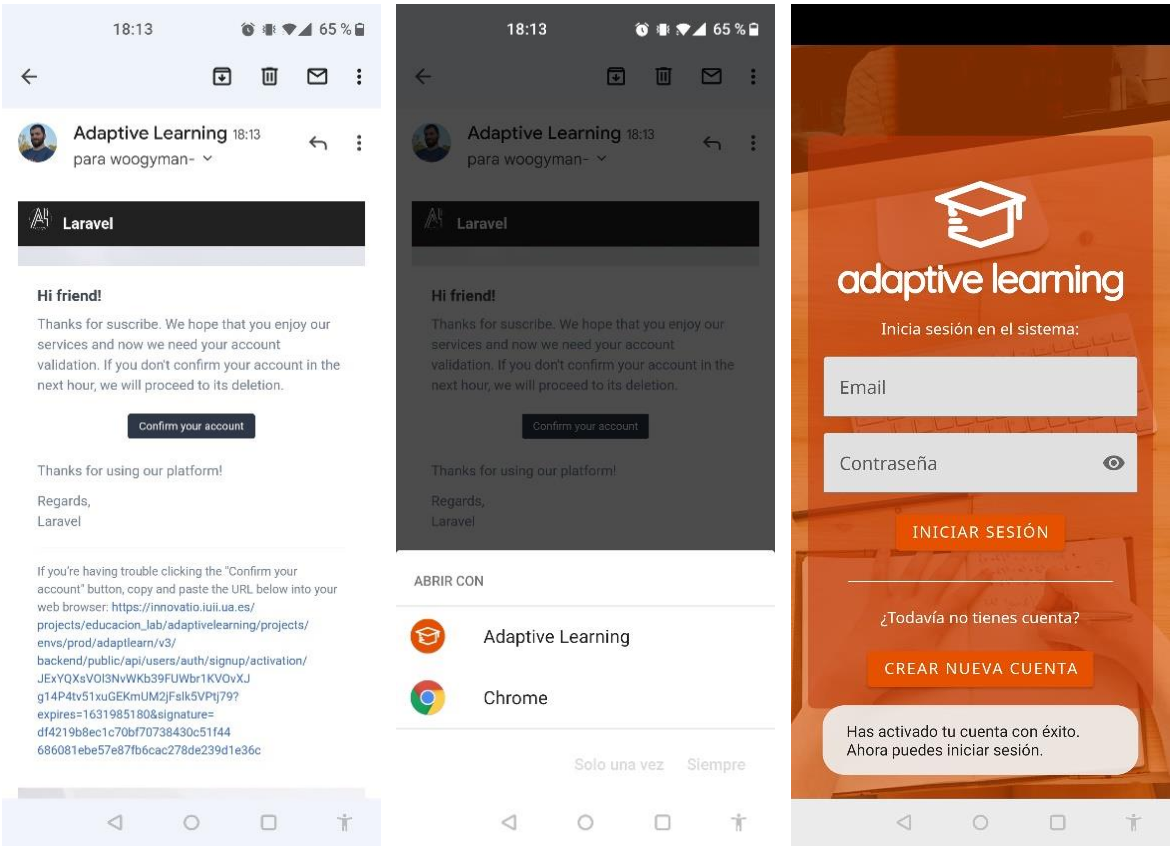

Ilustración 64: A3 - Email de confirmación / Abrir con Adaptive Learning / Confirmación de activación

Cursos

Ilustración 65: A3 - Vista de cursos (Listado de competencias)

La pantalla de los cursos (Ilustración 65) es la principal y la más compleja de todas, pues en ella sucede todo lo relativo al aprendizaje y la docencia de los cursos: la visualización del progreso en cada competencia, la realización y evaluación de actividades, etc.

La primera actividad de esta pantalla es **`CoursesActivity`**, y es la que obtiene la lista de los cursos del usuario, tanto aquellos en los que está como aprendiz como en los que está como docente. Para facilitar el almacenamiento y la administración de los datos relacionados con la IU de manera optimizada para los ciclos de vida se ha creado un *ViewModel* llamado `CourseViewModel` que, entre otras cosas, guarda el listado de cursos

estudiados y el listado de cursos con docencia, así como el curso que se haya seleccionado de ellos. Por lo tanto, la actividad `CoursesActivity` comprueba al inicio si se han descargado previamente los listados de cursos, y de no ser así hace peticiones a la API para obtenerlos. También se comprueba en este *ViewModel* si se ha seleccionado previamente uno de los cursos, y de ser así, crea y carga un `LearningCourseFragment` o un `TeachingCourseFragment` (dependiendo de si el usuario está como aprendiz o como docente en el curso seleccionado) a los que se les pasa el id del curso y el título. Una vez obtenidos todos los cursos, se actualiza la interfaz para mostrar en la barra el título del curso seleccionado. Si no se hubiera seleccionado ningún curso, se selecciona por defecto el primero de los cursos con aprendizaje, y si no los hubiera, el primero de los cursos con docencia.

Además, si el usuario dispone de más de un curso, se crea un *Bottom Sheet Dialog* (Ilustración 66) que aparecerá al pulsar en el título del curso en la barra. Este diálogo mostrará las listas de los cursos que está "cursando" e "impartiendo", y si selecciona uno de ellos, el diálogo se cierra y la actividad carga el *fragment* correspondiente de los anteriormente mencionados.

Ilustración 66: A3 - Vista de cursos (Selección de curso)

El *fragment* `LearningCourseFragment`, que se carga cuando el usuario selecciona un curso en el que se encuentra como aprendiz, comprueba en primer lugar si en el `CourseViewModel` hay algún curso seleccionado previamente. Si no lo hay, se hace la petición a la API solicitando todos los detalles del curso. En cuanto se obtiene esta información se carga el *fragment* que mostrará las diferentes competencias del curso y sus estados (Ilustración 67). Dependiendo de lo que el usuario haya elegido en la interfaz, esta información se visualizará en forma de listados de competencias o en forma de mapa de competencias (como un grafo dirigido acíclico). Los *fragments* que muestran esto son `CourseSkillsListFragment` y `CourseSkillsMapFragment` respectivamente. El usuario puede cambiar entre estas dos visualizaciones haciendo clic en el *toggle* que se muestra a la izquierda del título “Competencias”.

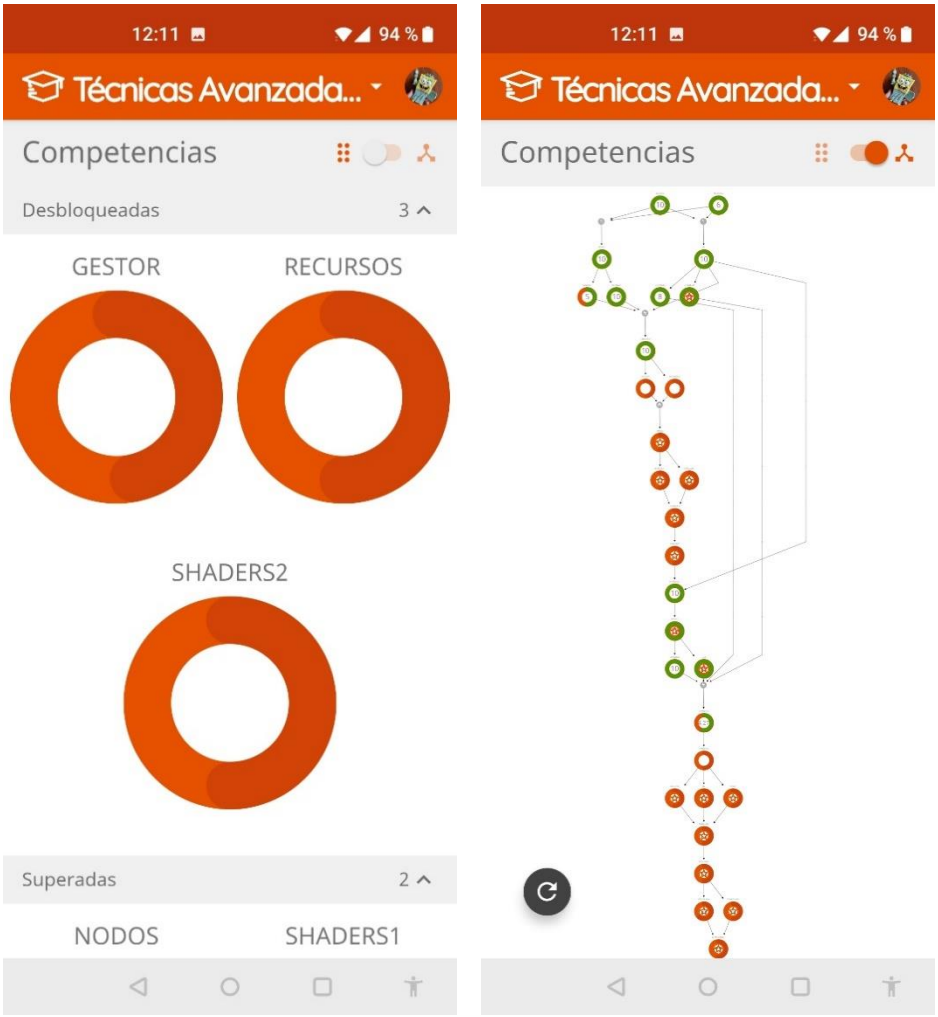

Ilustración 67: A3 - Vista de curso (Listado de competencias / Mapa de competencias)

En `CourseSkillsListFragment` se hace uso de varios `FlexboxLayout` para listar los círculos que representan las competencias según el estado de éstas para el usuario: desbloqueadas, comenzadas, superadas, completadas, bloqueadas y agotadas. En `CourseSkillsMapFragment` se hace uso de la librería `GraphView`, en concreto de la configuración Sugiyama, que permite representar el mapa de las competencias como un grafo dirigido.

El círculo que representa una competencia de un curso (Ilustración 68) es una vista que se ha creado para el proyecto, `SkillCircleView`, compuesta principalmente por dos vistas `FitChart` capaces de mostrar una rueda de progreso. El círculo completo (naranja claro) representa la puntuación máxima obtenible en la competencia, el progreso de la primera rueda (naranja oscuro) representa la puntuación mínima para aprobar la competencia y el progreso de la segunda rueda (amarillo) representa la puntuación obtenida por el aprendiz en la competencia. Este dato también se muestra como una cifra en el centro del círculo.

Ilustración 68: A3 - Vista de la competencia de un curso

Si en cualquiera de las dos visualizaciones de las competencias el usuario hace clic sobre el círculo de una de ellas se abre un *Bottom Sheet Dialog* (Ilustración 69) que muestra más información sobre la competencia y su estado, así como un botón para comenzar a practicarla que estará activo únicamente si la competencia se encuentra disponible.

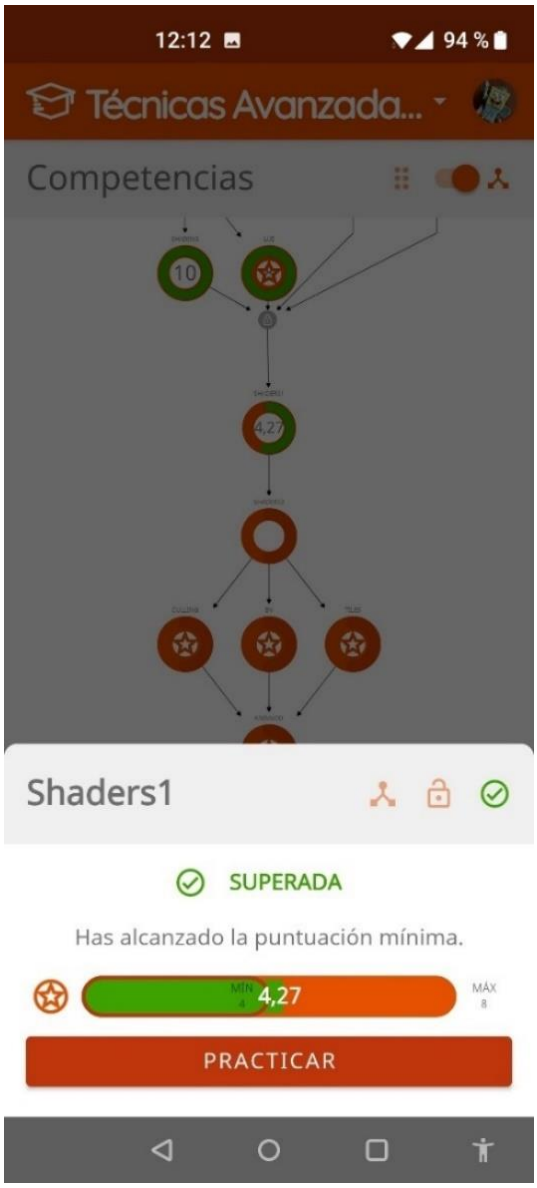

Ilustración 69: A3 - Vista de curso (Previsualización de competencia de curso)

En este diálogo se cargarán tres *fragments* diferentes cada vez, en función de la información sobre la competencia que desee conocer el usuario. El usuario podrá cambiar de uno a otro haciendo clic en los iconos que aparecen en la parte superior derecha del diálogo.

El primer *fragment* muestra el estado del progreso del aprendiz en la competencia: sin comenzar, sin superar, superada y completada (Ilustración 70 e Ilustración 71). Indica la puntuación que ha obtenido y muestra los umbrales mínimo y máximo de la competencia.

Ilustración 70: A3 – Vista del estado del progreso de una competencia (Sin comenzar / Sin superar)

Ilustración 71: A3 – Vista del estado del progreso de una competencia (Superada / Completada)

El segundo *fragment* muestra la disponibilidad de la competencia para el aprendiz: bloqueada en el tiempo, bloqueada por no cumplir una dependencia, agotada por haber realizado todas las actividades disponibles en la competencia o desbloqueada (Ilustración 72 e Ilustración 73). Si las hay, muestra también las dependencias con otras competencias que hacen que ésta esté bloqueada o desbloqueada.

Ilustración 72: A3 – Vista del estado de la disponibilidad de una competencia (Bloqueada en el tiempo / Bloqueada)

Ilustración 73: A3 - Vista del estado de la disponibilidad de una competencia (Agotada / Desbloqueada)

Por último, el tercer *fragment* (Ilustración 74) muestra aquellas competencias que dependen de la actual, si las hubiera, indicando qué puntuación se ha de obtener en esta para desbloquearlas.

Ilustración 74: A3 - Vista de competencias dependientes de una competencia

Al hacer clic en el botón activado de este diálogo se abrirá una nueva actividad, **`CourseSkillActivity`** (Ilustración 75), que se ocupa de cargar las actividades de aprendizaje para que el usuario las realice. Esta hace uso de otro *ViewModel*, `CourseSkillViewModel`, para almacenar, entre otras cosas, la competencia del curso, la actividad que está siendo realizada y la siguiente actividad a realizar.

`CourseSkillActivity` comprueba en primer lugar si se dispone de toda la información de la competencia del curso, y de lo contrario envía una petición a la API para obtenerla. Cuando se dispone de todos los datos, se actualiza la información de la interfaz, como por ejemplo, la barra de progreso de la competencia que se muestra en la parte superior. Después, se comprueba mediante el `CourseSkillViewModel` si se debe mostrar la pantalla de información de haber superado un umbral, o si se está en mitad de la realización de una actividad de aprendizaje y por lo tanto se debe mostrar. De lo contrario, no se hará nada hasta que el usuario haga clic en “Comienza a practicar la competencia”. En ese momento se envía una petición a la API solicitando la siguiente actividad a realizar para el aprendiz que seleccionará de las disponibles para él. Al obtenerla, se almacena en el `CourseSkillViewModel` como la actividad actual y se carga el *fragment* `CourseSkillActivityFragment`.

Este *fragment* detecta el tipo de actividad del que se trata y carga el *fragment* correspondiente a ese tipo. Por ahora únicamente se dispone de actividades tipo test, así que el *fragment* cargado es `ActivityTestFragment`. Además, también comienza a realizar peticiones a la API para crear o actualizar el log de “última conexión”

(LAST_CONNECTION), y así disponer en la base de datos del último momento en el que el usuario se encontró realizando la actividad de aprendizaje.

Ilustración 75: A3 - Vista de la competencia de un curso para un aprendiz

El *fragment* `ActivityTestFragment`, al inicio, comprueba en primer lugar si junto a la actividad de aprendizaje se han recibido cambios sin guardar que el aprendiz realizó anteriormente. Si es así, se le muestra una alerta preguntándole si desea cargar dichos cambios y se cargan o no según su respuesta. Si no se han cargado cambios sin guardar se comprueba si se ha recibido cambios que el usuario sí decidió guardar, y en ese caso se cargan. Seguidamente se procesa el fichero de la actividad, si es que tiene uno, de manera que si es un PDF lo descarga y lo muestra en una vista de tipo *PDFView*, si es un vídeo de *youtube* o de *vimeo* lo carga en una *WebView*, y si es una imagen en una *ImageView*. Por último, procesa las preguntas que forman el tipo test, añadiendo *fragments* a la vista en función del tipo de pregunta que sea.

Si es una pregunta de verdadero o falso, añade un *fragment* de la clase `ActivityTestTFFragment` (Ilustración 76). Este *fragment* crea un `Switch` por cada respuesta que forme parte de la pregunta. Si el usuario cambia un *switch*, el *fragment* le indica a `ActivityTestFragment`, a través del *ViewModel*, que ha de enviar una petición a la API para crear un log de "cambios en el progreso" (CHANGE_PROGRESS).

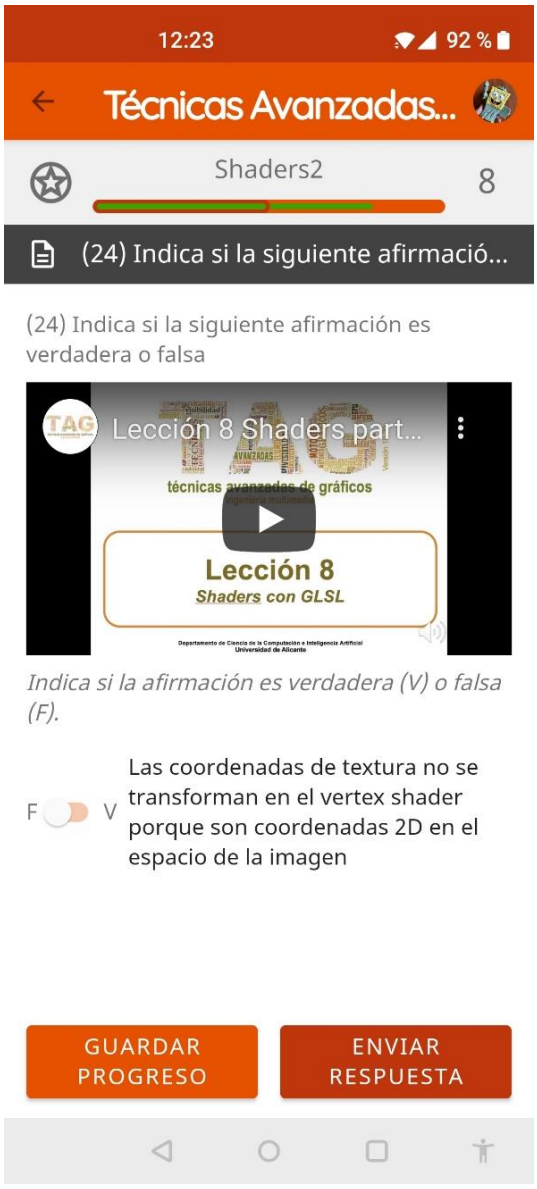

Ilustración 76: A3 - Vista de la realización de una actividad tipo test de Verdadero / Falso

Si es una pregunta de respuesta única, es decir, con una única respuesta correcta, añade un *fragment* de la clase `ActivityTestUniqueFragment` (Ilustración 77). En él se crea un `RadioButton` para cada respuesta a la pregunta. Si el usuario selecciona alguna de las

respuestas, el *fragment* le indica a `ActivityTestFragment` que ha de enviar una petición para crear un log de “cambios en el progreso” (CHANGE_PROGRESS).

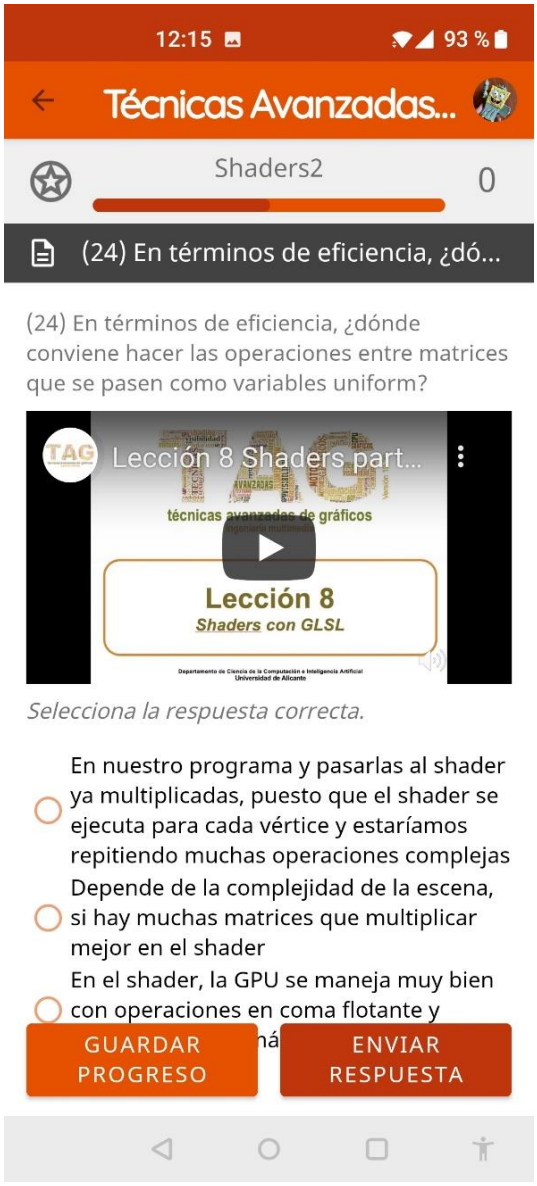

Ilustración 77: A3 - Vista de la realización de una actividad tipo test de respuesta única

Por último, si es una pregunta de respuesta múltiple, es decir, que puede tener más de una respuesta correcta, añade el *fragment* de la clase `ActivityTestMultipleFragment` (Ilustración 78). En él se crea un `CheckBox` por cada respuesta a la pregunta. Si el usuario marca o desmarca alguno de estos *checkboxes*, como en los anteriores casos, el *fragment* le indica a `ActivityTestFragment` que ha de enviar una petición para crear un log de “cambios en el progreso” (CHANGE_PROGRESS).

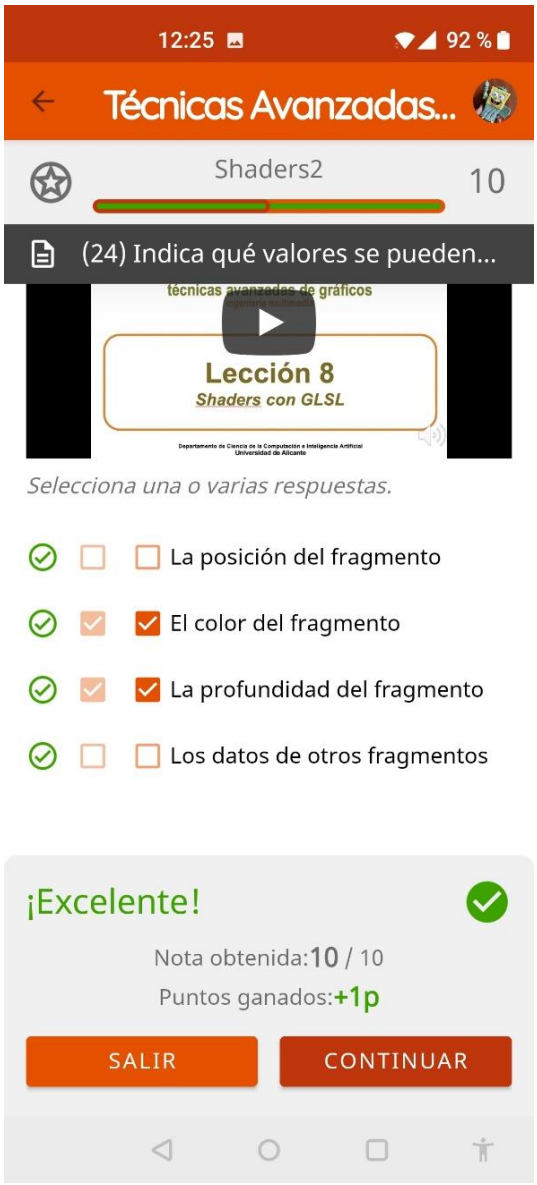

Ilustración 78: A3 - Vista de la realización de una actividad tipo test de respuesta múltiple

En todo momento, haciendo uso de los botones de `ActivityTestFragment`, el usuario puede decidir hacer un guardado de sus cambios realizados en la actividad de aprendizaje (guardando un log tipo SAVE) o enviar sus respuestas para que la actividad le sea evaluada.

Al realizar esta última acción, el sistema, además de guardar un log tipo FINISH, corregirá el ejercicio y devolverá en primer lugar la actividad con la solución y la nota obtenida y en segundo la siguiente actividad que el usuario podría hacer si desea continuar realizando actividades (si hay alguna disponible). Además, aquí se detecta si se

ha alcanzado la puntuación umbral mínima o máxima de la competencia o si ya no hay ninguna actividad disponible más.

Los datos de la actividad corregida se utilizan en `CourseSkillActivityFragment` para actualizar la puntuación y el progreso de la competencia del curso que se muestran en la parte superior y para crear una "tarjeta" en la parte inferior de la pantalla (Ilustración 79) que muestre la nota obtenida, así como los puntos que ha ganado al realizar la actividad. En esta tarjeta, si el usuario pulsa en el botón "Salir" se le devolverá a la pantalla del curso, y si pulsa en "Continuar" se le notificará a la actividad `CourseSkillActivity` que se desea pasar a la siguiente actividad de aprendizaje.

Ilustración 79: A3 - Vista del resultado obtenido en la realización de una actividad (0 puntos / 10 puntos)

`CourseSkillActivity`, al recibir esta notificación, comprueba en primer lugar si se acaba de alcanzar la puntuación umbral mínima o máxima de la competencia o si la competencia ha quedado agotada (porque ya no hay más actividades que el usuario pueda realizar), y de ser así se crea y carga el *fragment* que mostrará la información correspondiente, `CourseSkillThresholdReachedFragment` (Ilustración 80), antes de poder pasar a la siguiente actividad. Si no ha ocurrido nada de eso, se pasa directamente a dicha actividad.

Ilustración 80: A3 - Vista de umbral alcanzado (Competencia superada / Competencia completada)

Si en algún momento el usuario decide salir para dejar de practicar la competencia, se le devuelve a la actividad `CoursesActivity`, y en ella, el *fragment* con el curso estudiado,

`LearningCourseFragment`, en primer lugar actualiza el progreso de las diferentes competencias y después comprueba si se ha conseguido desbloquear nuevas competencias por haber sobrepasado algún umbral de dependencia. De ser así, muestra aquellas competencias recién desbloqueadas (Ilustración 81).

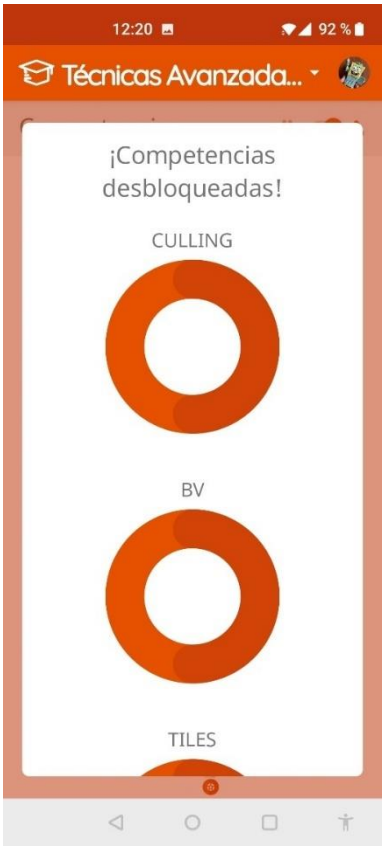

Ilustración 81: A3 - Vista de las competencias recién desbloqueadas

Por otro lado, si se selecciona un curso en el que el usuario tenga docencia del listado de cursos, el *fragment* que se carga es `TeachingCourseFragment`, que ofrece dos botones que permiten cambiar entre dos *fragments*: `EvaluatingTeachingCourseFragment` y `UsersTeachingCourseFragment`.

El primero de estos *fragments* se carga por defecto o pulsando en el icono de la lista con el *check* que aparece a la derecha del título “Docente”. En él se muestran las competencias del curso de manera similar a como se hacía para un curso en el que el usuario era el aprendiz (Ilustración 82). Las competencias por defecto no muestran progreso, pero se ofrece un selector de aprendiz en la parte inferior que permite al docente elegir a un usuario de los aprendices del curso para poder visualizar su progreso en las competencias.

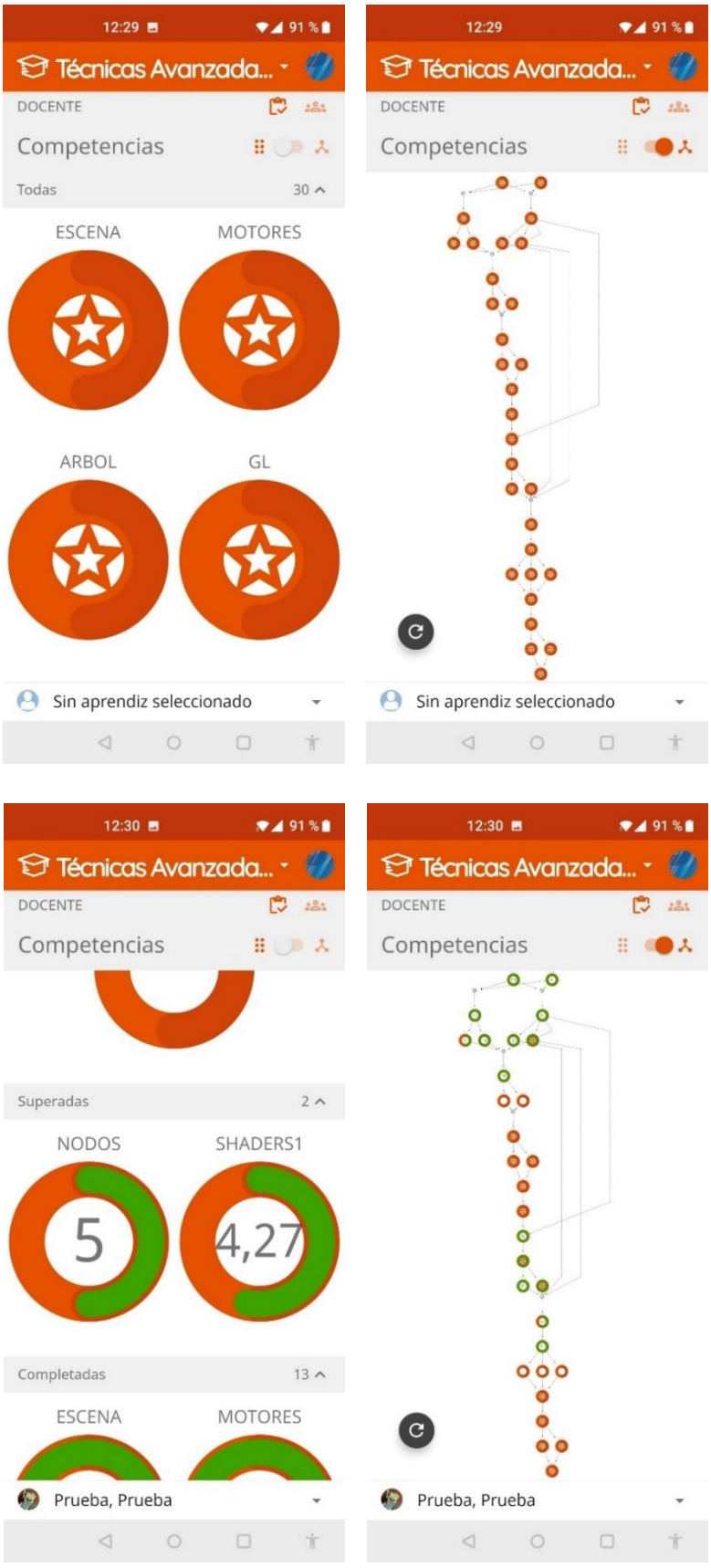

Ilustración 82: A3 - Vista de curso para docente (Listado / Mapa / Listado de aprendiz / Mapa de aprendiz)

Si hace clic en una de las competencias puede ver más información sobre el estado de esta para el aprendiz seleccionado (Ilustración 83).

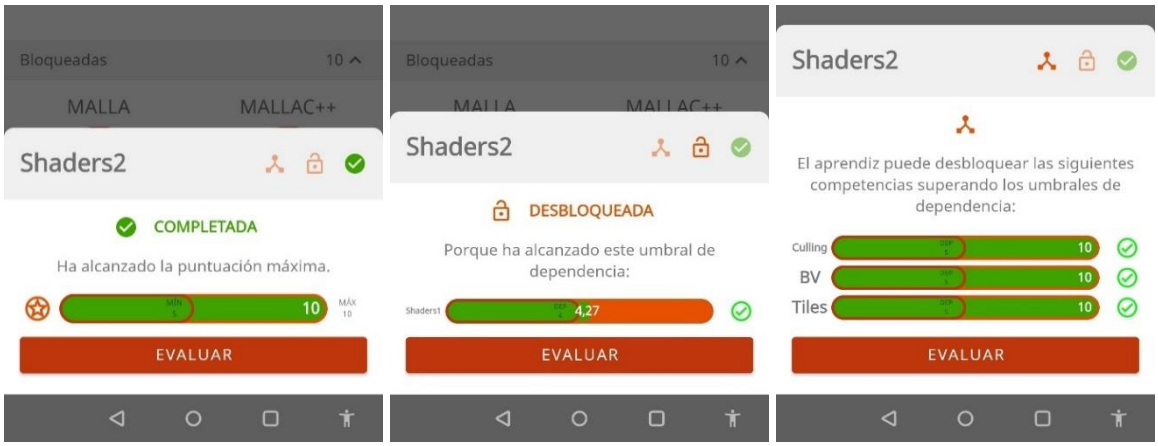

Ilustración 83: A3 - Vista de previsualización de la competencia de un aprendiz para docente (Progreso / Disponibilidad / Dependencias)

Además, si hace clic en el botón “Evaluar” podrá acceder a la actividad que muestra la evaluación de una competencia, **`CourseSkillTeachingActivity`** (Ilustración 84). En ella se envía una petición a la API para obtener todas las actividades de aprendizaje relacionadas con la competencia elegida. Si se ha seleccionado un aprendiz previamente, se recibirán además las notas que ha obtenido en cada una de las actividades.

Ilustración 84: A3 - Vista de la competencia de un aprendiz para un docente

Se puede seleccionar una de las actividades para visualizar su contenido con las soluciones y, si se ha seleccionado un aprendiz, con las respuestas dadas por éste. Esto se muestra en la actividad **`ActivityTeachingActivity`** (Ilustración 85).

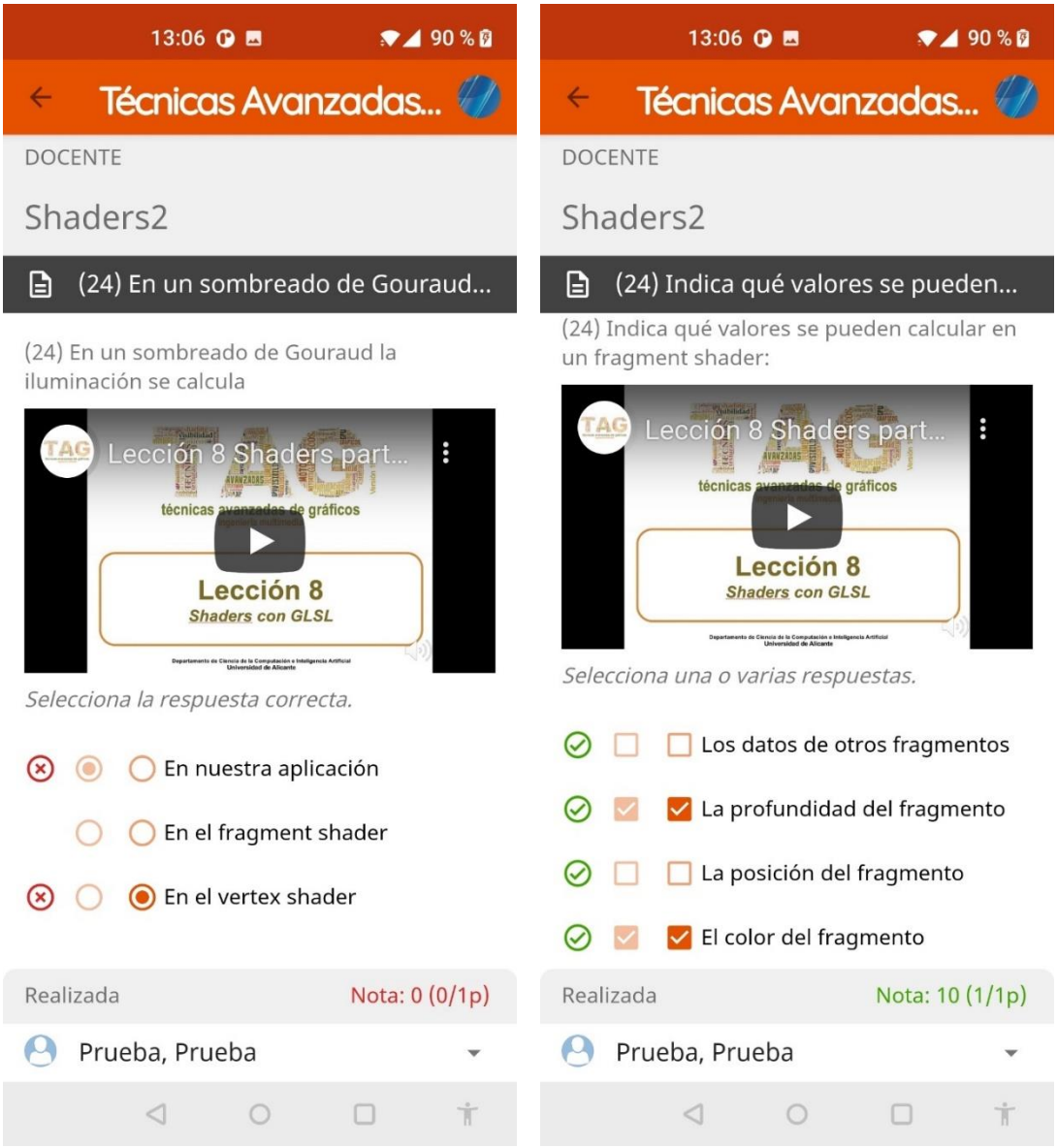

Ilustración 85: A3 - Vista de una actividad realizada por un aprendiz para un docente

Por último, si desde el `TeachingCourseFragment` se hace clic en el icono de personas que aparece a la derecha del título “Docente” se cargará el *fragment* de los usuarios del curso, `UsersTeachingCourseFragment` (Ilustración 86). En él se mostrará una lista con los usuarios aprendices y otra con los usuarios docentes.

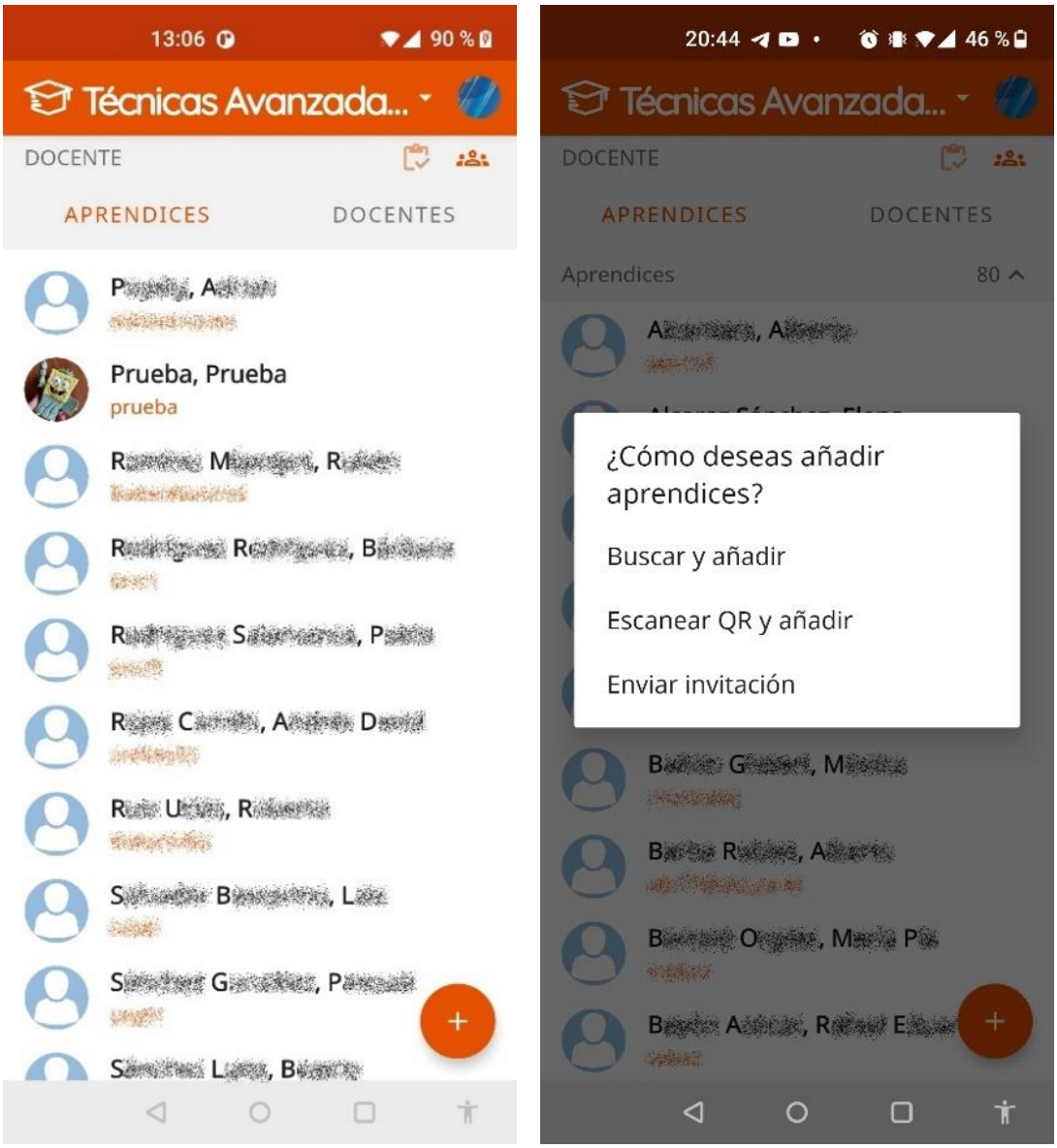

Ilustración 86: A3 - Vista de los usuarios de un curso (Listado de aprendices / Añadir aprendices)

Se puede hacer clic en el botón flotante con el “+” para añadir usuarios (aprendices o docentes). Al hacerlo, se muestra un diálogo en el que el usuario puede elegir la manera de añadir usuarios entre “Buscar y añadir”, “Escanear QR y añadir” y “Enviar invitación”.

Si elige “Buscar y añadir” se abre la actividad `RelationsEditActivity` (Ilustración 87) en la que puede realizar una búsqueda de usuarios, seleccionar todos aquellos que desee añadir pulsando en sus avatares y finalmente confirmar la adición pulsando en el botón flotante verde con el icono del *check*.

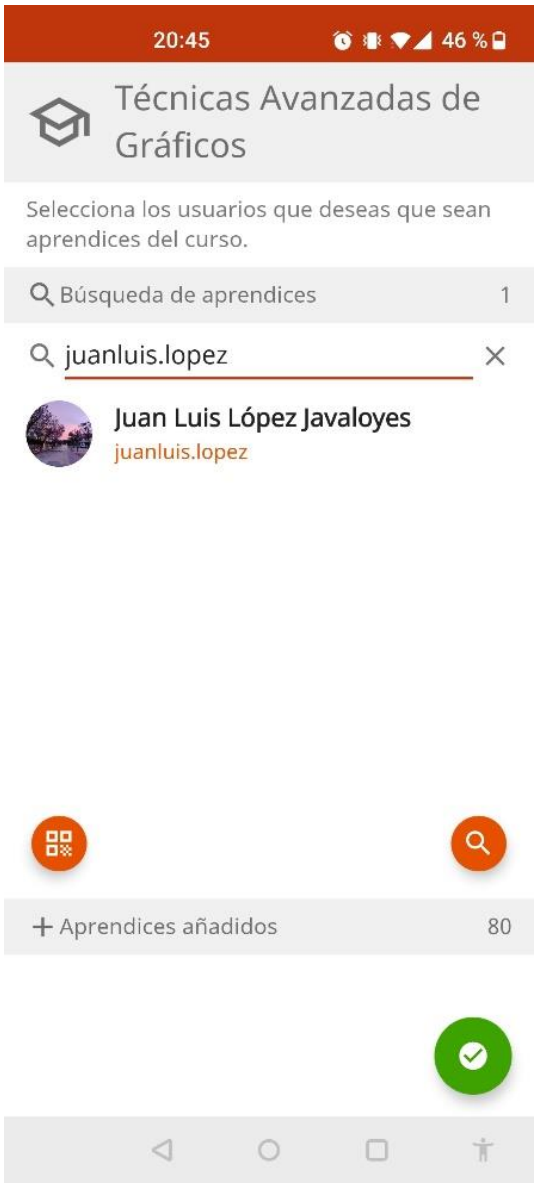

Ilustración 87: A3 - Vista de adición de aprendices

Si elige “Escanear QR y añadir” se lanza un *intent* que abre la cámara con el objetivo de realizar el escaneado de códigos QR (Ilustración 88). En cuanto se encuentra uno, el código se lee extrayendo los datos del usuario al que hace referencia. Dichos datos se devuelven y se le indican al docente a la vez que se le pide confirmación para añadirlo. Si lo confirma, se envía una petición a la API enviándole los datos del nuevo usuario a añadir. Si todo ha ido bien, la lista se recargará, mostrando el nuevo usuario añadido.

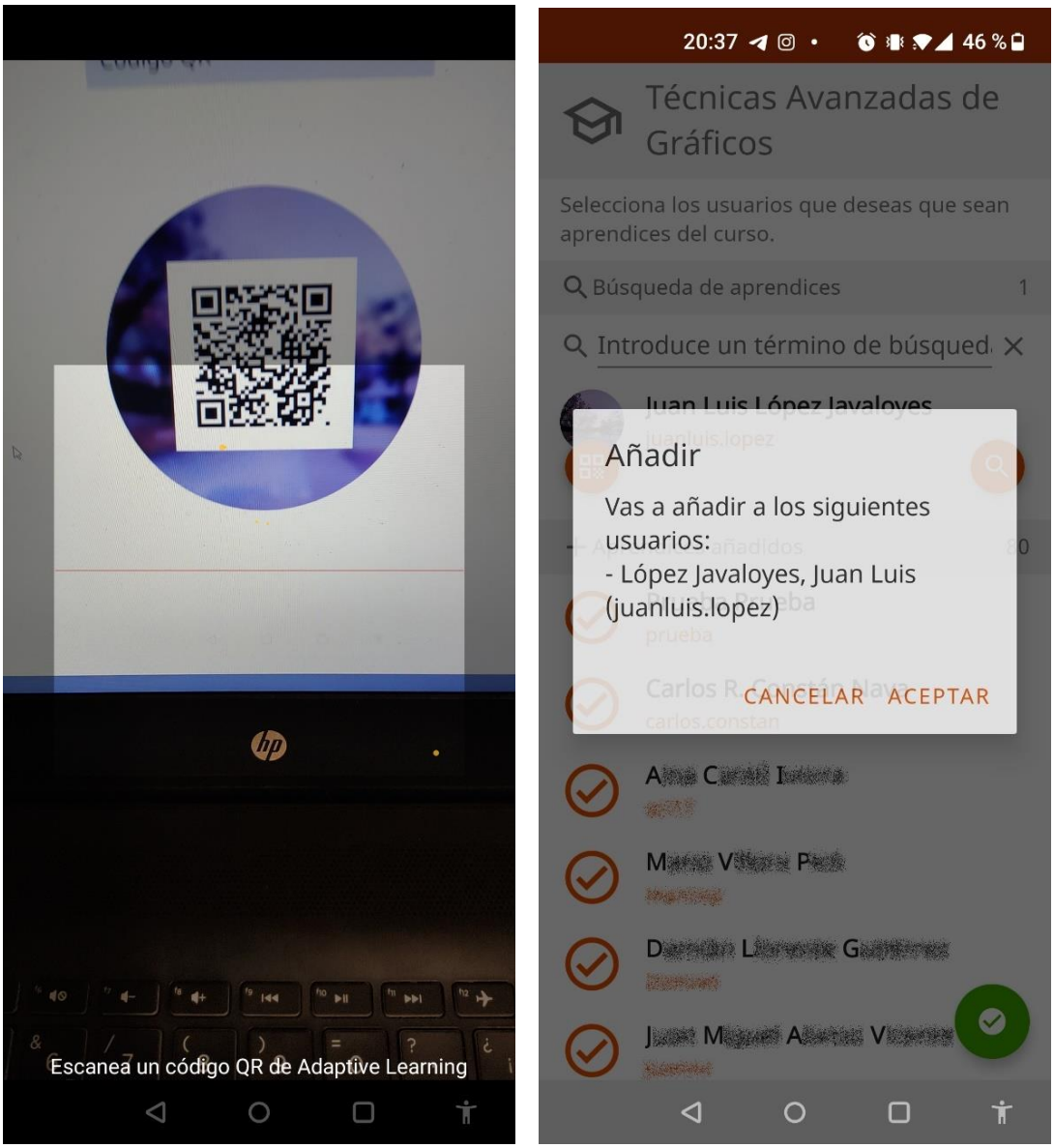

Ilustración 88: A3 - Vista de escaneo de QR / Aviso para añadir usuario

Diseñador

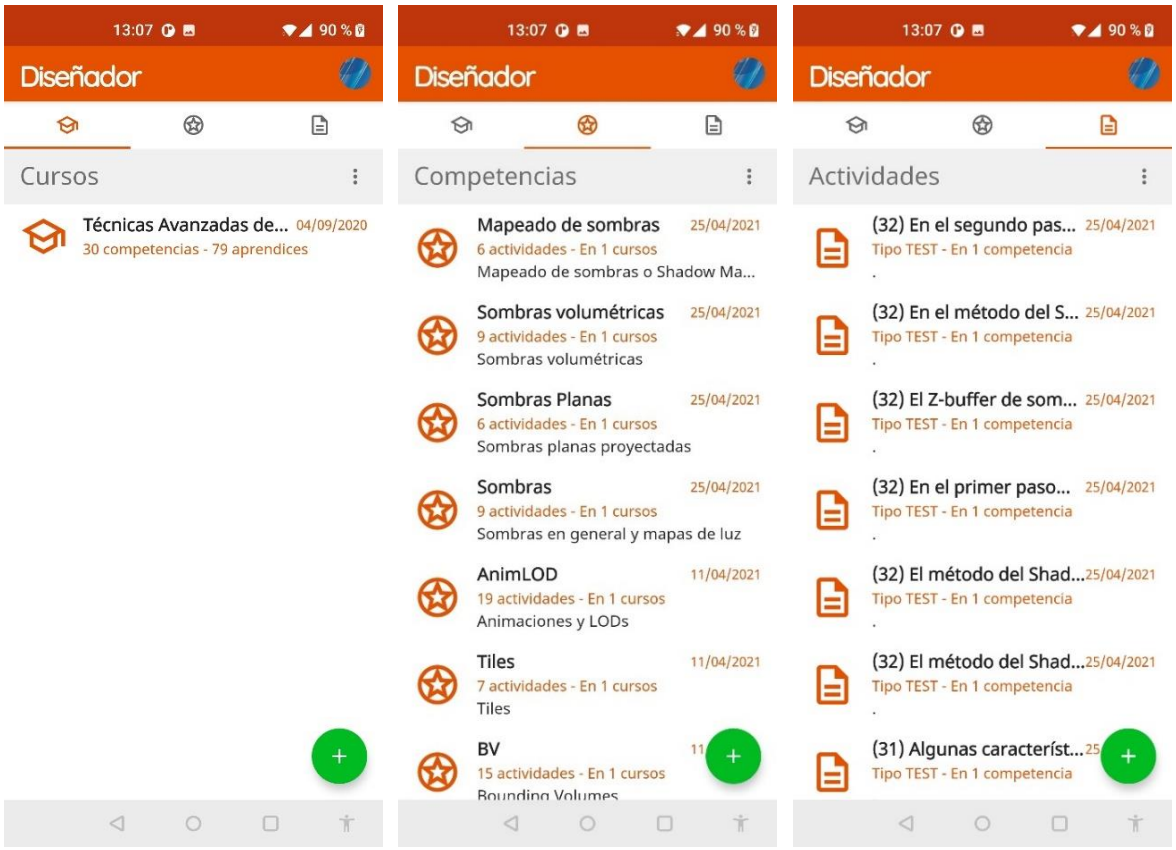

Ilustración 89: A3 - Vista del diseñador (Listado de cursos / Listado de competencias / Listado de actividades)

La pantalla del diseñador (Ilustración 89) es donde sucede toda la creación de contenidos de la plataforma. Aunque por el momento tan solo se ha implementado la parte de creación/edición de competencias, también tendrá lugar aquí la creación/edición de actividades y de cursos.

La actividad en la que sucede la visualización de las diferentes listas de elementos diseñados es en **`DesignerActivity`**, que hace uso del *fragment* `TabsFragment` para manejar las tres posibles pestañas diferentes.

En este *fragment* se comprueba en primer lugar qué pestañas estarán disponibles para el usuario. Actualmente estarán disponibles las tres pestañas, pero quizá en el futuro se desee limitar los elementos creables/editables por el usuario, y desde aquí se podrá elegir qué pestañas mostrar. El adaptador para las páginas de las pestañas es el encargado de distinguir si se trata de un dispositivo de gran tamaño, tipo tablet, o si es un dispositivo menor. De ser una tablet, se carga el *fragment* `MasterDetailHostFragment` que mostrará una vista tipo *Master-Detail* (Ilustración 90) con los elementos de la página

que se esté mostrando en ese momento, y si no, se carga el *fragment* `MasterHostFragment` que únicamente muestra la parte del *Master*, es decir, el listado de elementos.

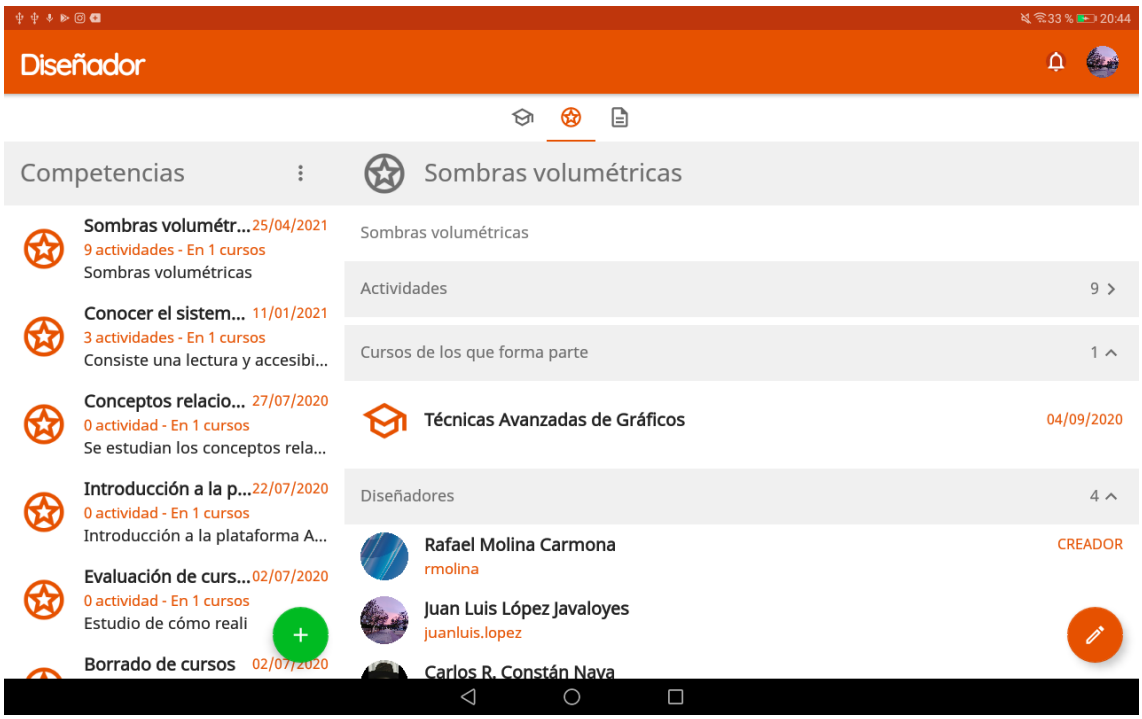

Ilustración 90: A3 - Vista Master-Detail del diseñador en tablet

`MasterFragment` es el que se ocupa de cargar el listado de elementos en función de la pestaña de la que se trata. Si es la pestaña de las competencias, cargará el *fragment* `SkillListFragment`, si es la pestaña de las actividades, cargará el *fragment* `ActivityListFragment`, y si es la pestaña de los cursos, el *fragment* cargado será `DesignedCourseListFragment` (Ilustración 89).

Dado que tan solo se ha llevado a cabo la implementación completa del listado de competencias, se omite la explicación de los otros dos listados, que seguirán una implementación semejante.

En `SkillListFragment` se hace uso del *ViewModel* `ElementListViewModel` para guardar el estado de la vista en cada momento. Entre otras cosas, se almacenan los elementos cargados, si la lista se encuentra en "modo selección múltiple" o en "modo búsqueda", así como el filtro aplicado a la lista si lo hubiera o los ids de los elementos seleccionados.

Por lo tanto al principio, cuando se ha creado la vista del *fragment*, se comprueba si estaba activado alguno de los modos y se modifica la interfaz en consecuencia. Seguidamente se inicializa la lista, comprobando primero si se tenía algún elemento cargado en el *ViewModel*, y de no ser así, cargar los elementos haciendo la correspondiente petición a la API.

Para el listado de competencias en sí se hace uso de un *fragment* genérico, `ListFragment`, creado para cualquier tipo de elemento, junto a un adaptador de *RecyclerView* llamado `ListRecyclerViewAdapter`, que implementa un *scroll* infinito para detectar cuándo cargar más elementos porque se ha hecho *scroll* hacia abajo. También detecta cuándo el usuario toca en el icono de un elemento de la lista. Cuando esto ocurre, `SkillListFragment` se encarga de marcar el elemento como seleccionado o no seleccionado y activar el modo selección (Ilustración 91).

Ilustración 91: A3 - Vista del diseñador (modo selección de lista)

En este modo, aparece el icono de una papelera junto al número de elementos seleccionados. Si lo pulsamos, aparecerá una alerta para confirmar que se desean eliminar los elementos seleccionados (Ilustración 92). Si confirmamos, se envía una petición a la API solicitando el borrado de las competencias cuyos ids se le han enviado. Si todo va bien, la lista se recarga para mostrarla sin los elementos eliminados.

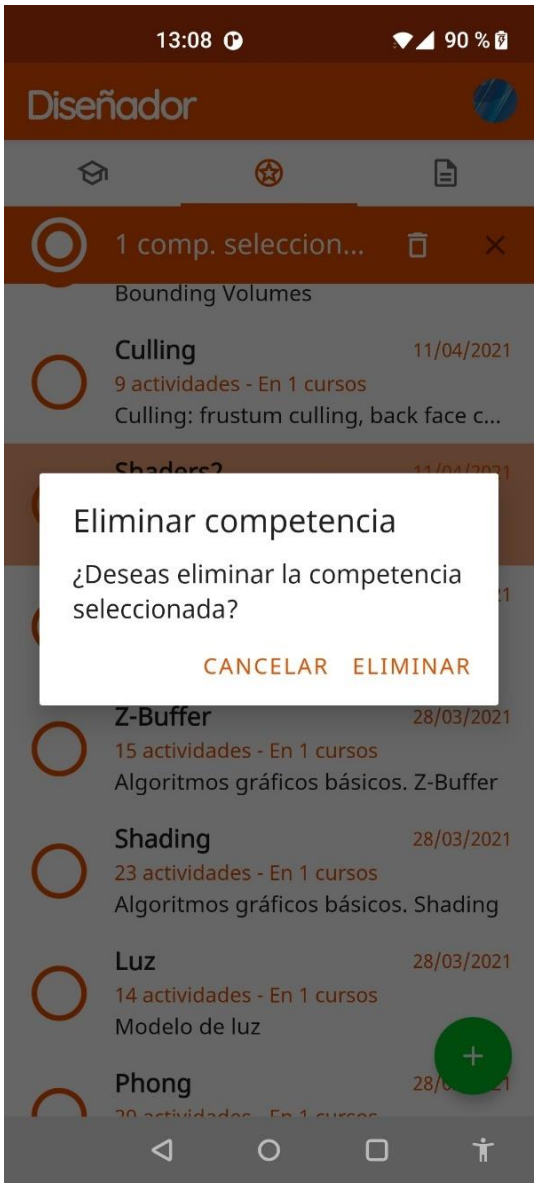

Ilustración 92: A3 - Vista del diseñador (Confirmación de eliminación de elementos)

Si se hace clic en el botón de más opciones que se muestra a la derecha del título “Competencias” se abre un menú con opciones, desde el que se puede activar el “modo búsqueda”, el “modo selección” o acceder a crear una nueva competencia (Ilustración 93). Esta última acción también se puede llevar a cabo haciendo clic en el botón flotante con el “+”.

Ilustración 93: A3 - Vista del diseñador (Menú)

Si se activar el “modo búsqueda” (Ilustración 94) se muestra un campo de texto donde antes aparecía el título “Competencias”. Si se escribe algo en dicho campo y se pulsa “intro” en el teclado o se hace clic en el icono de la lupa, se añadirá la cadena de texto al filtro para buscarla en el título de las competencias. Después se envía la petición a la API, que hará la búsqueda y devolverá las competencias que coincidan con el término introducido. Se puede salir de este modo y del modo selección pulsando en el icono con la “x” que aparece a la derecha.

Ilustración 94: A3 - Vista de diseñador (modo búsqueda)

Si se hace clic en uno de los elementos (fuera de su icono), se cargará la vista del elemento. Esto se hará diferente dependiendo de si el dispositivo es una tablet o no. Si lo es, se cargará el *fragment* `DetailFragment` en el espacio del *fragment* `MasterDetailHostFragment` reservado para el mismo, pasándole el tipo de elemento del que se trata. Si no es una tablet, se abrirá la actividad `DetailActivity`, que a su vez cargará en su interior el `DetailFragment`.

`DetailFragment` es el que se ocupa de cargar el *fragment* de la vista del elemento en concreto, en función de la pestaña en la que se encuentra el usuario (es decir, el tipo de elemento). Si es la pestaña de las competencias, cargará el *fragment* `SkillFragment`, si

se trata de la pestaña de las actividades o de los cursos todavía no se ha implementado la carga de ningún otro *fragment*, pero en el futuro cargará `ActivityFragment` y `DesignedCourseFragment`, respectivamente.

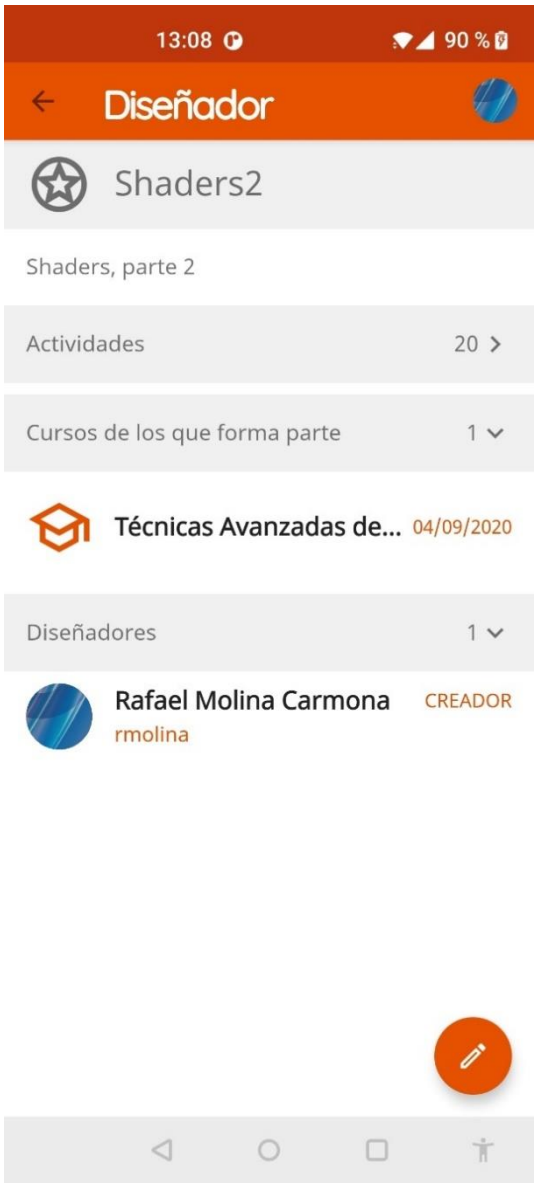

Ilustración 95: A3 - Vista de competencia diseñada

`SkillFragment` (Ilustración 95) lo primero que hace es comprobar si se ha cargado ya la competencia y en ese caso mostrar su información, y si no, enviar una petición a la API solicitando la competencia diseñada por el usuario. De ésta en la interfaz se muestra su título, su descripción y listas de otros elementos relacionados, como por ejemplo las actividades vinculadas, los cursos en los que se encuentra y los usuarios que pueden

editarla. Si el número de elementos de una de estas listas es superior a 5, la lista se muestra en una actividad nueva, `SkillRelationsActivity`.

Se puede acceder a editar la competencia haciendo clic en el botón flotante con el icono del lápiz. Esto abrirá la actividad **`SkillEditActivity`** (Ilustración 96), en el que encontramos un campo para editar el título y otro para la descripción, y dos botones que llevan a la actividad **`RelationsEditActivity`** (Ilustración 96) para añadir/quitar actividades de aprendizaje relacionadas con la competencia y diseñadores que pueden editarla. Si se hace algún cambio en estos datos, el botón flotante con el icono de guardar se activa, y si es pulsado se envía una petición a la API, pasándole todos los datos de la competencia, para actualizarla.

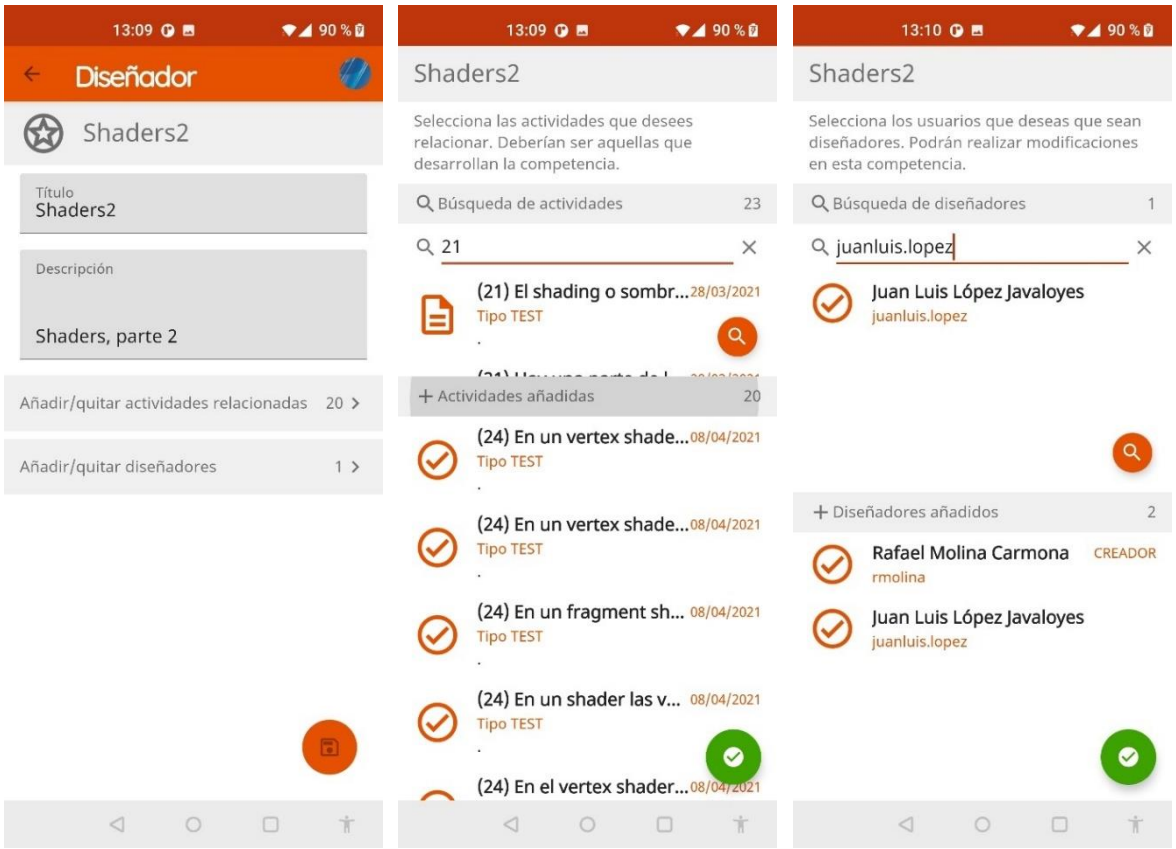

Ilustración 96: A3 - Vista de edición de competencia / Vista para añadir/quitar actividades / Vista para añadir/quitar diseñadores

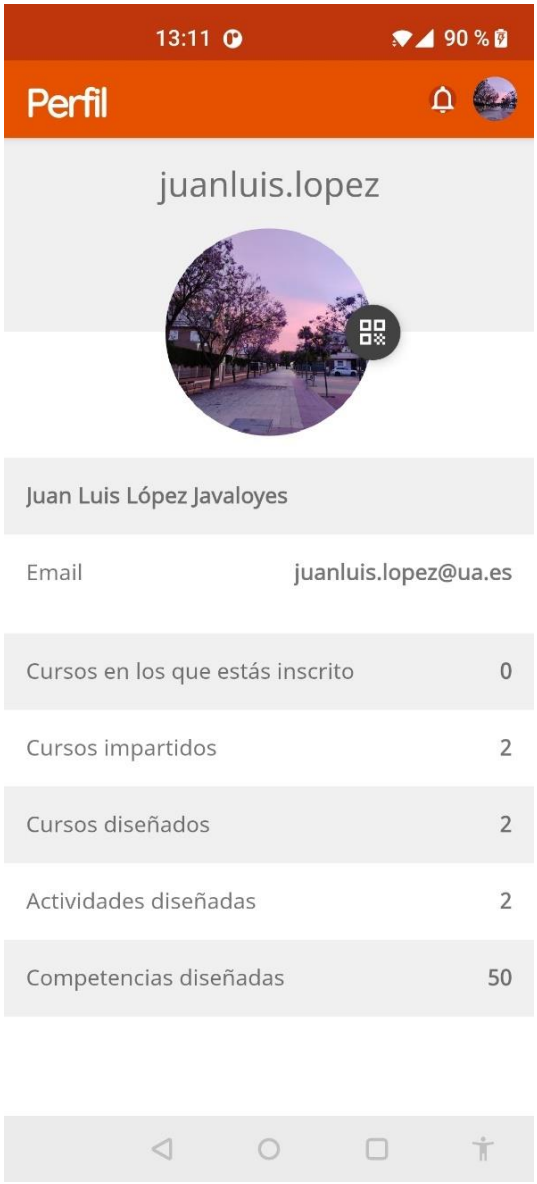

Ilustración 97: A3 - Vista del perfil

El perfil del usuario (Ilustración 97) es la pantalla en la que éste puede ver su propia información. La actividad encargada de mostrarla es **`ProfileActivity`**, y lo primero que hace es solicitar la información completa del usuario enviando una petición a la API. Cuando la recibe, actualiza los datos, que son: su nick de usuario, su imagen de avatar, su nombre completo, su email y una serie de contadores del número de cursos en los que está inscrito, cursos que imparte, cursos que diseña, actividades y competencias que diseña.

Si se hace clic en el botón que aparece junto a la imagen de avatar se mostrará el código QR de su usuario (Ilustración 98), que podrá ser utilizado por un docente para añadirlo a un curso como aprendiz o docente, tal y como hemos visto anteriormente. Además, si hace clic en el icono de compartir, se iniciará un *intent* de tipo *SEND* que permitirá al usuario seleccionar una aplicación a través de la cual que enviar la imagen de su código QR (Ilustración 98).

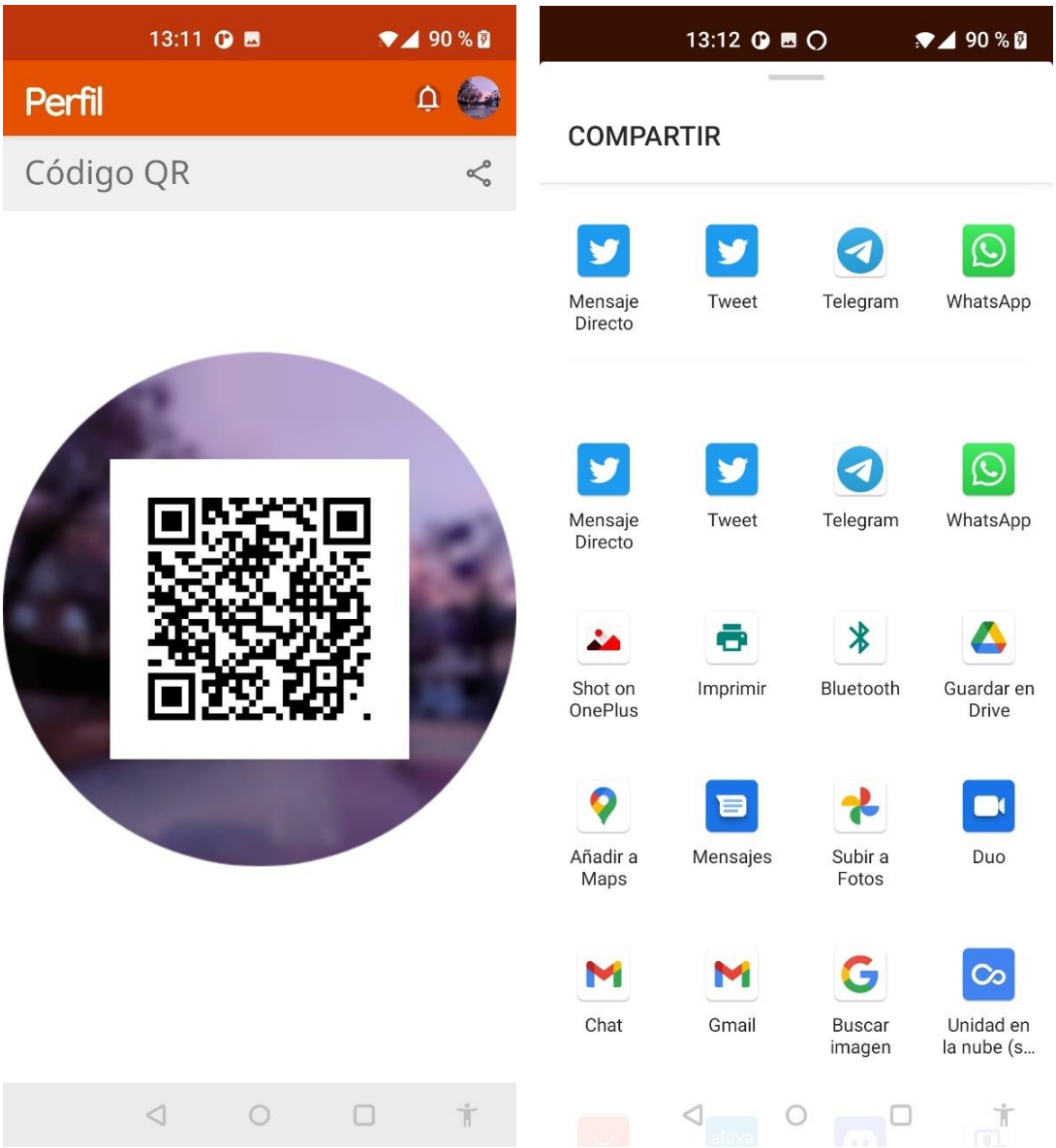

Ilustración 98: A3 - Vista del QR del usuario / Compartir imagen del QR

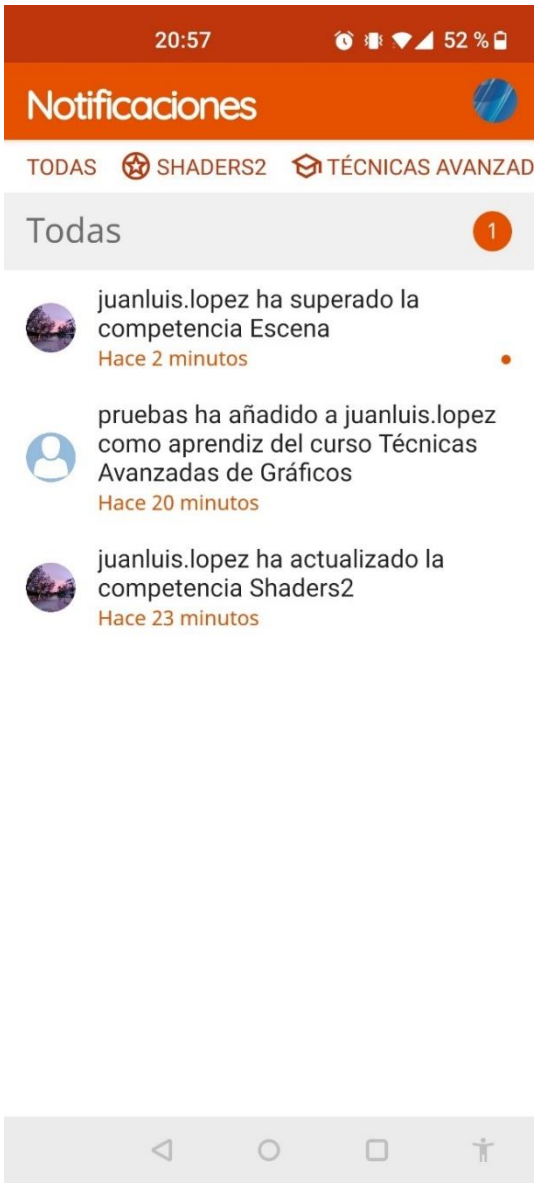

Ilustración 99: A3 - Vista de notificaciones

La pantalla de notificaciones (Ilustración 99) es donde se muestran las notificaciones del usuario. La actividad encargada de realizar esta operación es **`NotificationsActivity`**, y utiliza el *fragment* `ListFragment` para mostrar el listado de notificaciones del usuario, manejar el *scroll* infinito, etc. En primer lugar se comprueba si ya se han cargado las notificaciones, y si no ha ocurrido, se hace la petición a la API solicitándolas.

El número de notificaciones nuevas se muestra en un contador naranja, y en el listado se distinguen de las antiguas por el punto naranja que aparece a la derecha. Se puede tocar sobre una notificación para marcarla como vista o hacer un toque largo sobre una para

marcarla como no vista. En estos casos se envía una petición a la API solicitando cambiar el estado de la notificación.

Además, las notificaciones vienen organizadas por temática, según el elemento al que hacen referencia (Ilustración 100). Se puede pasar de una a otra temática haciendo clic en los botones que aparecen en la parte superior.

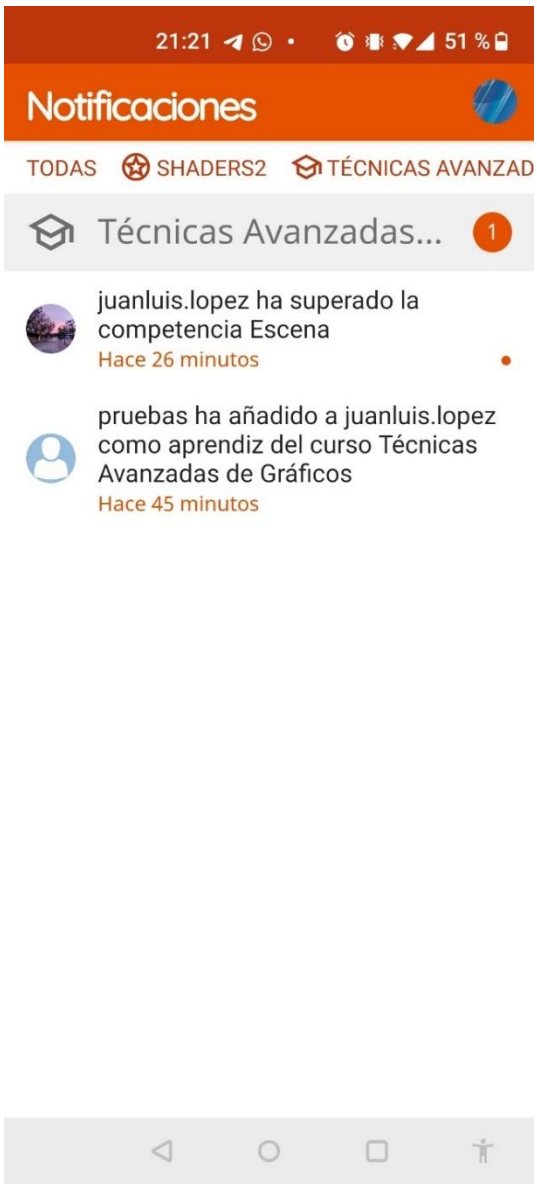

Ilustración 100: A3 - Vista de las notificaciones del curso Técnicas Avanzadas de Gráficos

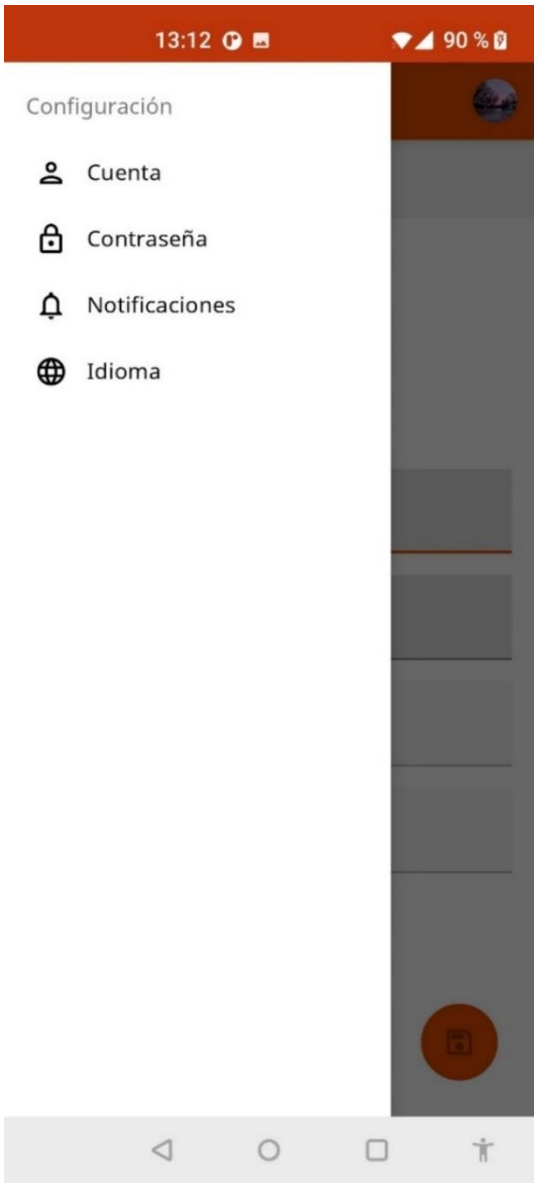

Ilustración 101: A3 - Vista de configuración (Menú)

La pantalla de configuración es donde el usuario puede modificar las opciones de la app y sus preferencias. La actividad en la que se maneja esto es **`SettingsActivity`** (Ilustración 101), que es la que carga uno u otro *fragment* dependiendo de la sección de la configuración que elija ver el usuario. El usuario puede seleccionar la sección a través de un menú dentro de un `DrawerLayout`.

El *fragment* que primero se carga, por defecto, es el de la configuración de la cuenta, `SettingsAccountFragment` (Ilustración 102), y en él el usuario puede modificar su nombre y apellidos, así como su imagen de avatar. Si se hace clic en el botón con icono

del lápiz que hay junto al avatar, el *fragment* crea e inicia un *intent* de tipo "`ACTION_GET_CONTENT`" para obtener contenido, en este caso, una imagen, pudiendo hacerlo desde la cámara o desde el explorador de archivos (Ilustración 102).

Cuando el usuario considere que ha realizado los cambios pertinentes, podrá pulsar en el botón flotante con el icono de guardar, y entonces se enviará la petición a la API solicitando la actualización de estos datos, y pasándole la imagen si hiciera falta. Por supuesto, los campos de nombre y apellidos se validan antes de poder enviar los cambios.

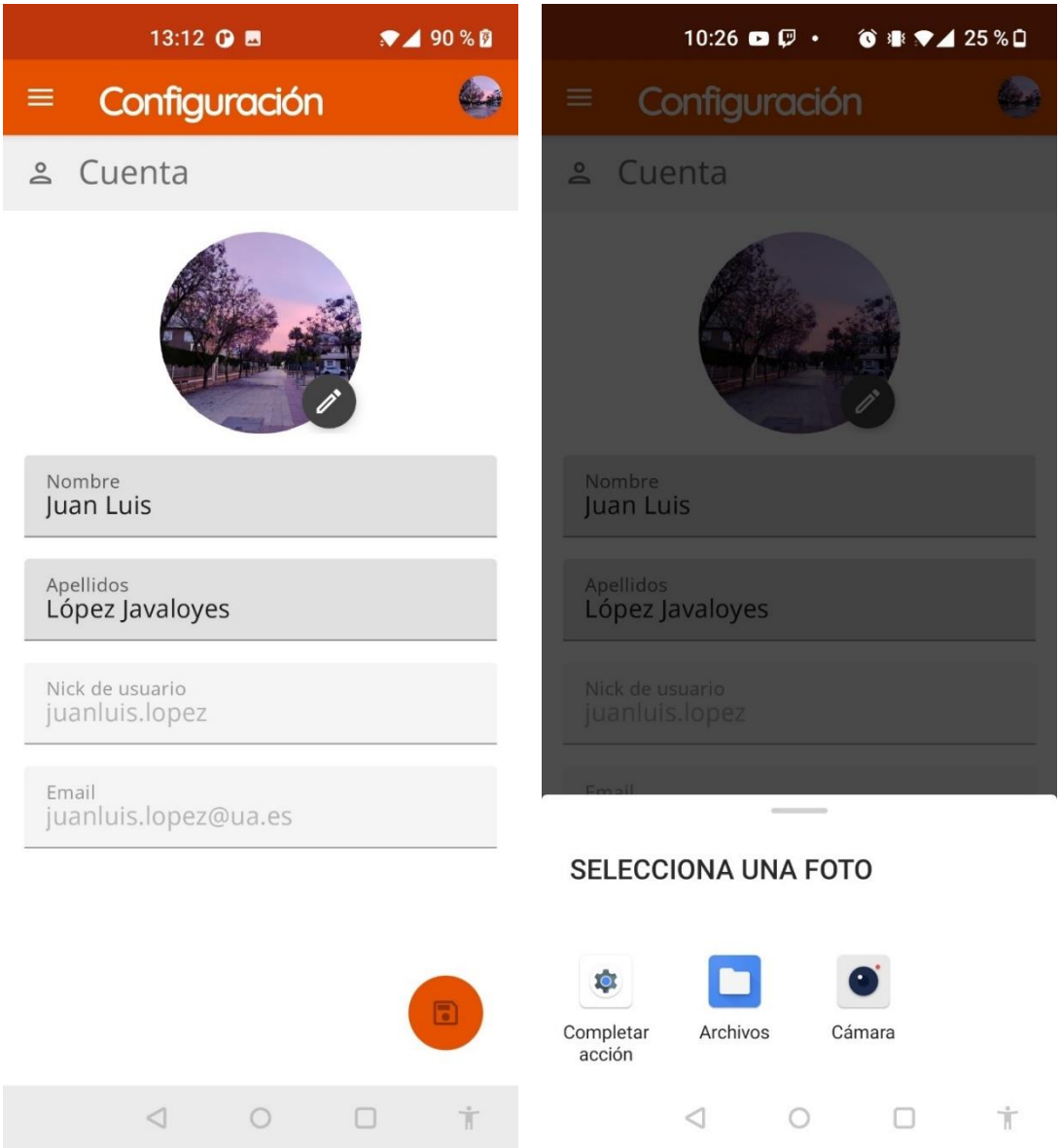

Ilustración 102: A3 - Vista de configuración de la cuenta / Selección de foto de avatar

Si en el menú se selecciona “Contraseña”, el *fragment* cargado es aquel en el que el usuario puede modificar su contraseña, `SettingsPasswordFragment` (Ilustración 103). En él se muestran tres campos: en el primero ha de introducir su contraseña actual, en el segundo la nueva y en el tercero repetir la nueva. Los campos se validan al pulsar el botón flotante de guardar, y si todo es correcto, se envía la petición a la API.

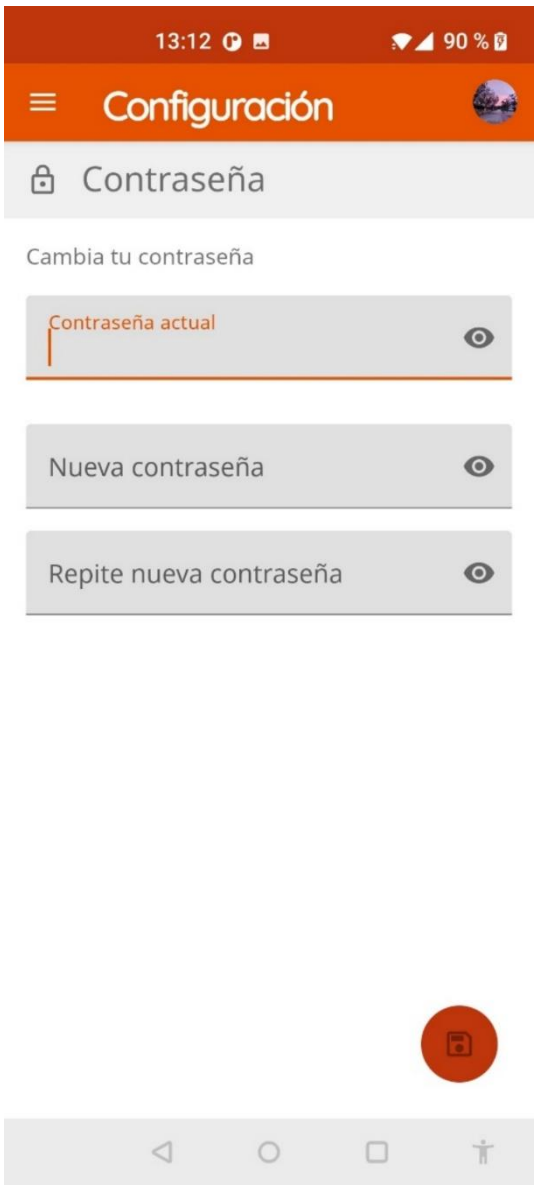

Ilustración 103: A3 - Vista de configuración de contraseña

Por otra parte, si en el menú se selecciona “Notificaciones”, el *fragment* que se carga es `SettingsNotificationsFragment` (Ilustración 104). En él se visualiza un listado con los diferentes tipos de notificaciones que puede haber en la aplicación, mostrado como un árbol desplegable, y, junto a cada tipo, un *toggle* para indicar si se quiere activar o

desactivar esas notificaciones y un icono de una campana que puede estar activada o desactivada en función de si se quieren recibir notificaciones *PUSH* de ese tipo. Por lo tanto, si se tienen las notificaciones de un tipo activadas (el *toggle* en modo ON) pero las alertas desactivadas (la campana desactivada) se mostrarán las notificaciones de dicho tipo en la pantalla de “Notificaciones” pero no se recibirá ninguna notificación *PUSH* de estas.

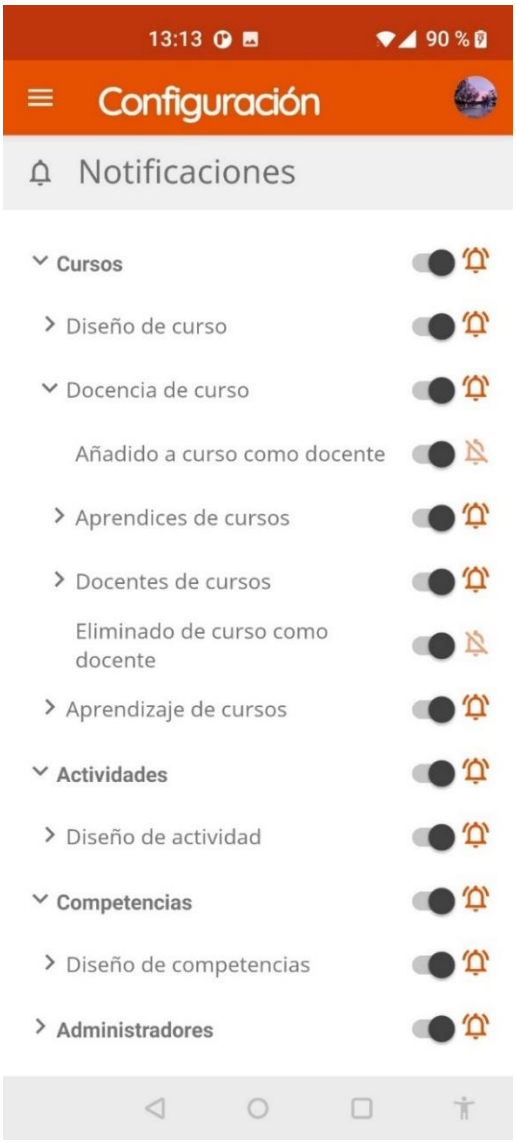

Ilustración 104: A3 - Vista de configuración de notificaciones

Por último, si en el menú se selecciona la opción “Idioma”, `SettingsLanguageFragment` (Ilustración 105) será el *fragment* cargado. En él se mostrará un desplegable con los posibles idiomas en los que se encuentra la aplicación, que por el momento son español e inglés.

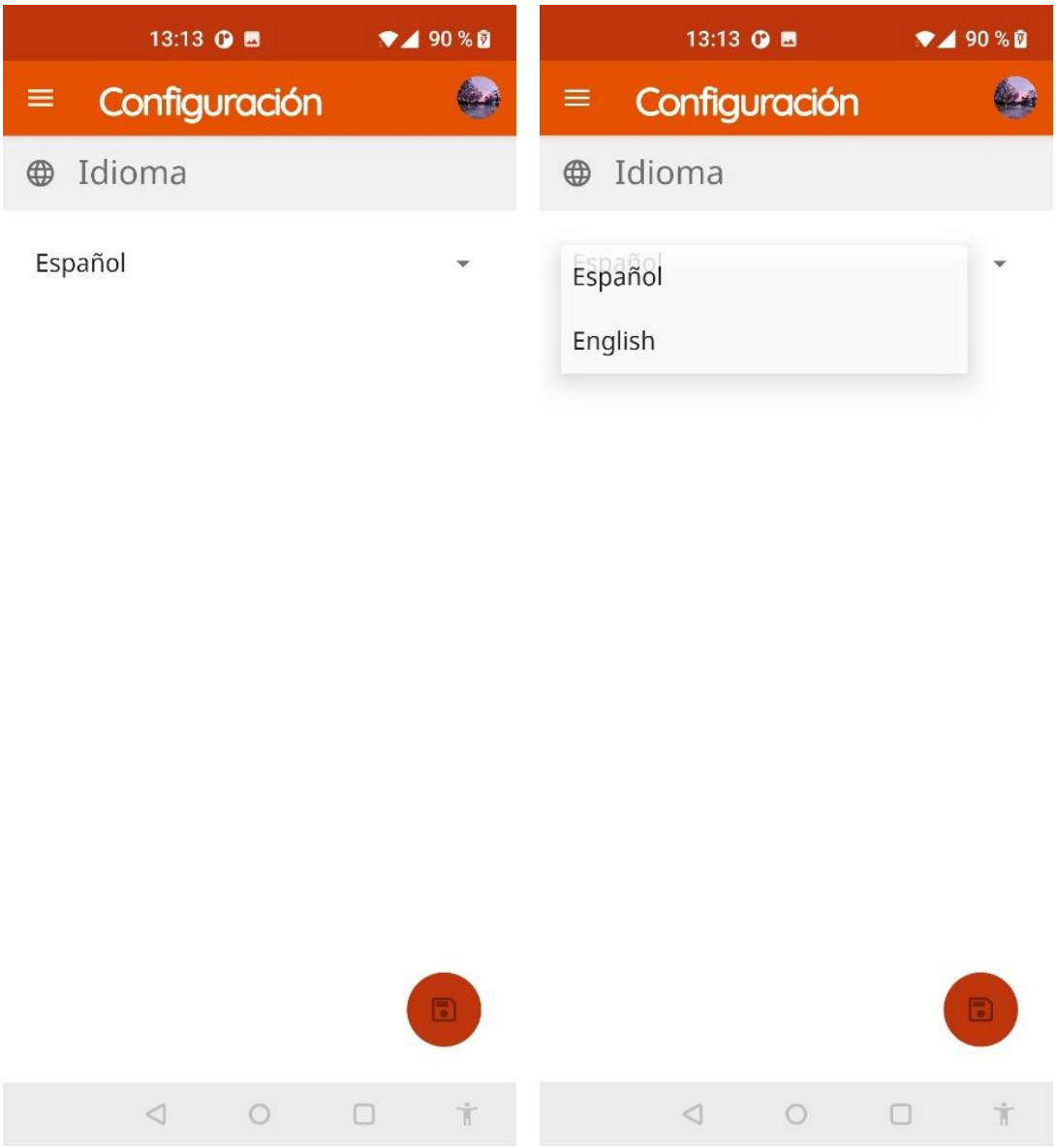

Ilustración 105: A3 - Vista de configuración del idioma

Si el usuario cambia de idioma y pulsa el botón de guardar, la petición se enviará a la API y la actividad se recargará, mostrando los textos en el idioma elegido (Ilustración 106).

Cabe destacar que todas las configuraciones que se hacen en la app se aplican también en la versión web, por lo que, si se cambia de idioma, los textos aparecerán en dicho idioma en cualquiera de los dispositivos por los que acceda a la plataforma, incluida la versión web de la misma.

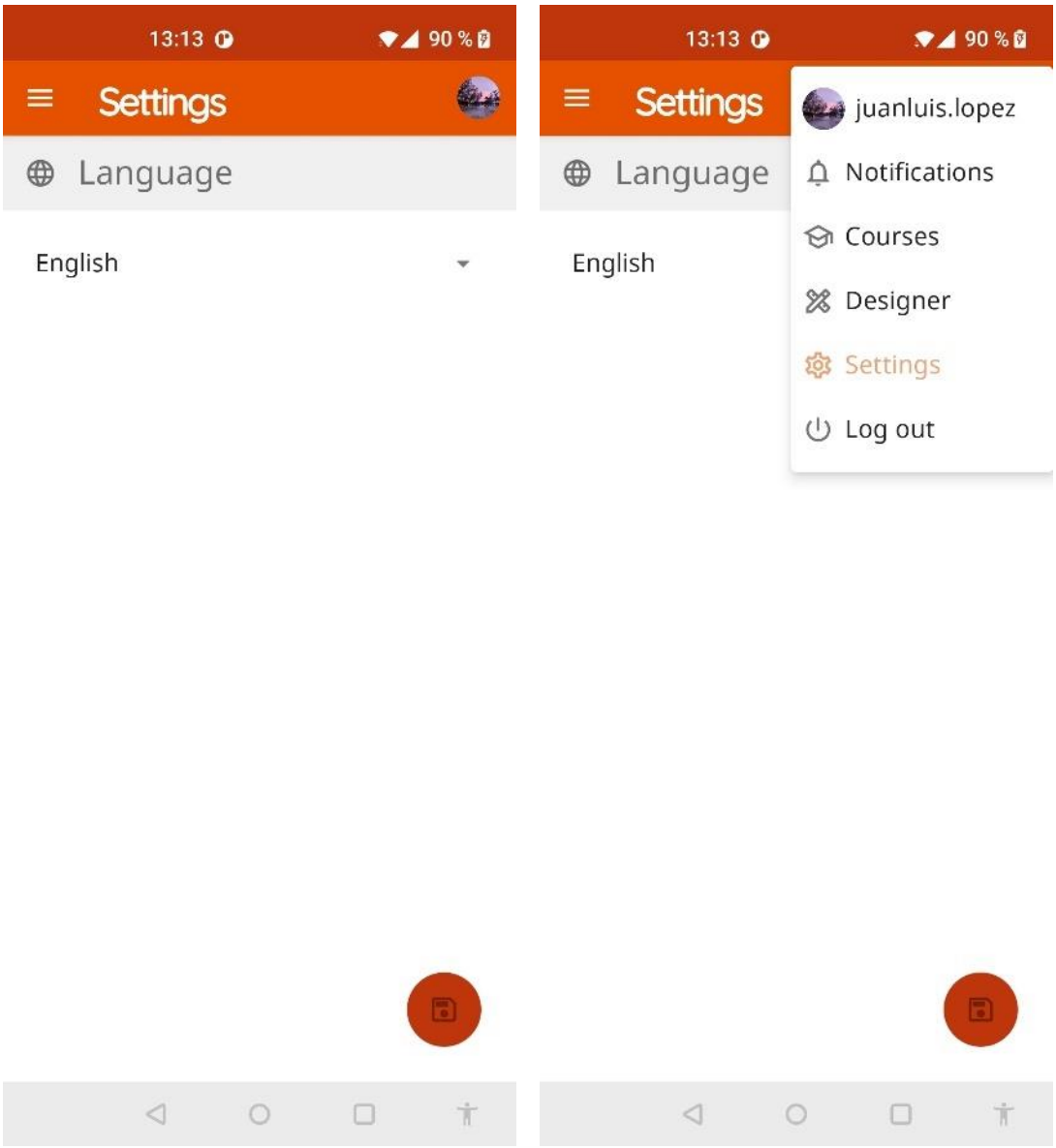

Ilustración 106: A3 - Vista de configuración de idioma con textos en inglés